intel

APPLICATION NOTE

AP-133

April 1982

Designing Memory Systems for Microprocessors Using the Intel 2164A and 2118 Dynamic RAMs

William H. Righter
Joseph P. Altnether

Order Number: 210431-001

PREFACE

This application note has been developed to provide the memory system designer with a detailed description of microprocessor memory system design using Intel Dynamic RAMs, the 16K 2118, 64K 2164A, and the 8203 Dynamic RAM Controller. The 8086 bus interface to memory components is described and three major examples are presented and analyzed — ranging from simple to complex: the simple solution, the 5 MHz No-WAIT State and the 10 MHz No-WAIT State systems. To assist the designer, complete logic schematics, timing diagrams and system design considerations are also included in this application note.

1 INTRODUCTION

Matching the correct RAM to microprocessor application requirements is fundamental to effective product design. A good understanding of the advantages and disadvantages of each technological approach and device type will enable a memory system designer to best choose the product that provides the optimal benefit for his particular design objective.

Two basic types of random access memories (RAMs) have existed since the inception of MOS memories: static RAMs (SRAMs) and dynamic RAMs (DRAMs). Where highest performance and simplest system design is desired, the static RAM can provide the optimum solution for smaller memory systems. However, the dynamic RAM holds a commanding position where large amounts of memory and the lowest cost per bit are the major criteria.

The major attributes of dynamic RAMs are low power and low cost — a direct result of the simplicity of the storage cell. This is achieved through the use of a single transistor and a capacitor to store a single data bit (Figure 1).

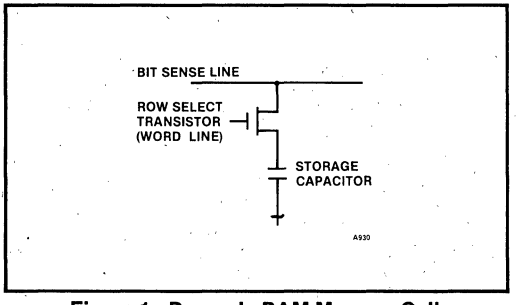


Figure 1. Dynamic RAM Memory Cell

The absence or presence of charge stored in the capacitor equates to a one or a zero respectively. The capacitor is in series with the transistor eliminating the need for a continuous current flow to store data. In addition, the input buffers, the output driver and all the circuitry in the RAM have been designed to operate in a sequentially clocked mode, thus consuming power only when being accessed. The net result is low power consumption. Also, a single transistor dynamic cell as compared to a four or six-transistor cell of a static RAM, occupies less die area. This results in more die per wafer.

Because the manufacturing cost of a wafer is fixed, more die per wafer translate into lower cost. For example, assume a wafer costs $250 to manufacture. Yielding 250 die per wafer means each die costs one dollar. But, if only 125 die are yielded, the cost per die is two dollars. The rationale of the quest for smaller die size is obvious; the simple dynamic memory cell fulfills this quest.

Unfortunately, the simple cell has a drawback: the capacitor is not a pure element and it has leakage. If left alone, leakage current would cause the loss of data. The solution is to refresh the charge periodically. A refresh cycle reads the data before it degrades too far and then rewrites the data back into the cell. RAM organization is tailored to aid the refresh function. As an example, the Intel® 2164A 64K RAM is organized internally as four 16K RAM arrays, each comprised of 128 rows by 128 columns. Consequently the row address accesses 128 columns in each of the four quadrants. However, let's concern ourselves with only one quadrant. Prior to selection, the bit sense line was charged to a high voltage. Via selection of the word line (row addresses) 128 bits are transferred onto their respective bit lines. Electrons will migrate from the cell onto the bit line destroying the stored charge. Each one of the 128 bit lines has a separate sense amplifier associated with it. Charge on the bit line is sensed, amplified and returned to the cell. Each time the RAM clocks in a row address, one row of the memory is refreshed. Sequencing through all the row addresses within 2 ms will keep the memory refreshed.

In spite of the advantages of minimal cost per bit and low power, the dynamic RAM has often been shunned in microprocessor systems. Up until now, dynamic RAMs have required a good deal of complicated circuitry to support the refresh requirements, and associated timing and interfacing needs. Circuitry for arbitration of simultaneous data and refresh requests, for example, has posed significant design problems. These requirements all add to the component count and system overhead costs, both in design and implementation.

The development of the Intel family of dynamic RAM controllers has brought a new level of design simplicity to dynamic RAM memory systems. These new devices include the solutions to the problems of arbitration, timing, and address multiplexing associated with dynamic RAMs.

This application note describes two basic memory systems employing the use of the Intel® 2164A and 2118 dynamic RAMs in conjunction with the Intel® 8203 Dynamic RAM Controller and the Intel 2164A, 64K dynamic RAM with a high speed TTL controller.

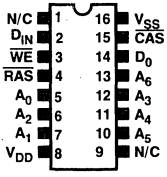
1.1 2118 16K RAM

The Intel 2118 is a high performance 16,384 word by 1 bit dynamic RAM, fabricated on Intel's n-channel HMOS technology. The Intel 2118 is packaged in the industry standard 16-pin DIP configuration, and only requires a single +5V power supply (with ±10% tolerances) and ground for operation, i.e., V_{DD} (+5V) and V_{SS} (GND). The substrate bias voltage, usually

designated V_{BB}, is internally produced by a back bias generator. The single +5V power supply and reduced HMOS geometries result in lower power dissipation and higher performance.

1.1.1 2118 DEVICE DESCRIPTION

The 2118 pin configuration and performance ratings are shown in Figure 2. Note that pins 1 and 9 are N/C (no-connects). This allows for future expansion up to 256K bits in the same device (package). For a rigorous device description, refer to AP-75, "Application of the Intel 2118 16K Dynamic RAM."



	2118-10	2118-12	2118-15
Access Time (ns)	100	120	150
Cycle Time (ns)	235	270	320
Operating Current (mA)	27	25	23
Standby Current (mA)	2	2	2

A931

Figure 2. Intel® 2118 Pinout

1.2.2 2118 ADDRESSING

Fourteen addresses are required to access each of the 16,384 data bits. This is accomplished by multiplexing the addresses onto seven address input pins. The two 7-bit address words are sequentially latched into the 2118 by the two TTL level clocks: Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). Noncritical timing requirements allow the use of the multiplexing technique while maintaining high performance. For example, a wide t_{RCD} window ($\overline{RAS}$ to $\overline{CAS}$ delay) allows relaxation of the timing sequence for $\overline{RAS}$, address change, and $\overline{CAS}$ while still permitting a fast t_{RAC} (Row Access Time).

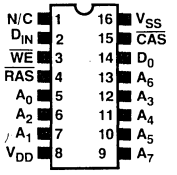
Data is stored in a single transistor dynamic storage cell. Refreshing is required for data retention and is accomplished automatically by performing a memory cycle (read, write or refresh) at all row addresses every 2 milliseconds.

1.2 2164A 64K RAM

The Intel 2164A is a high performance 65,536 word by 1 bit dynamic RAM, fabricated on Intel's advanced HMOS-D III technology. The 2164A also incorporates redundant elements. Packaged in the industry standard 16-pin DIP configuration, the 2164A is designed to operate with a single +5V power supply with ±10% tolerences. Pin 1 is left as a no-connect (N/C) to allow for future system upgrade to 256K devices. The use of a single transistor cell and advanced dynamic RAM circuitry enables the 2164A to achieve high speed at low power dissipation.

1.2.1 2164A DEVICE DESCRIPTION

The 2164A is the next generation high density dynamic RAM from the 2118 +5V, 16K RAM. The 2164A pin configuration and performance ratings are shown in Figure 3. For a detailed device description, refer to AP-131, "Intel 2164A 64K Dynamic RAM Device Description."



	2164A-15	2164A-20
Access Time (ns)	150	200
Cycle Time (ns)	260	330
Operating Current (mA)	55	45
Standby Current (mA)	5	5

A932

Figure 3. Intel® 2164A Pinout

1.2.2 2164A ADDRESSING

Sixteen address lines are required to access each of the 65,536 data bits. This is accomplished by multiplexing the 16-bit address words onto eight address input pins. The two 8-bit address words are latched into the 2164A by the two TTL level clocks: Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). Noncritical timing requirements allow the use of the multiplexing technique while maintaining high performance.

Data is stored in a single transistor dynamic storage cell. Refreshing is required for data retention and is accomplished automatically by performing a memory cycle (read, write or refresh) on the 128 combinations of A_0 through A_6 (row addresses) during a 2 ms period. Address input A_7 is a "don't care" during refresh cycles. Thus, designing a system for 256 cycle refresh at 4 ms in a distributed mode automatically provides 128 cycle refresh at 2 ms and a more universal system design.

1.3 Compatibility of the 2118 and the 2164A

In 2118 memory systems designed for upgradability, it is now possible to take advantage of the direct upgrade

path to the 2164A. The common pinout and similarities in A.C. and D.C. operating characteristics of most systems make this upgrade easy and straightforward. A simple jumper change to bring the additional multiplexed address into the memory array, a check for proper decoupling, and the replacement of the 2118's with 2164A's usually completes the job. In the two sections that follow, both device and system level compatibility issues are examined, key parameters are compared, and implications discussed. A data sheet for each device should be handy to aid in understanding the following material.

1.3.1 DEVICE COMPATIBILITY

Both the 2118 and 2164A are packaged in the industry standard 16-pin DIP. Observation of the device's pinout configurations shows that the only difference is the additional multiplexed address address input on pin 9 of the 2164A. This extra input is required to address the additional memory within. Notice the N/C (no connect) on pin 1 of the 2164A. This allows for another direct upgrade path to the 256K DRAM device, with pin 1 used as the next address input. The first and most obvious specifications to compare are the speed and cycle times. Clearly, when discussing compatibility and upgradability the same speed devices must be examined. A glance at the respective data sheets shows that the 2118-15 and the 2164A-15 are the current devices available that are speed and cycle time compatible, and further discussion will center on these two specific device types.

1.3.1.1 D.C. and Operating Characteristics

Both the 2164A and the 2118 function in the same temperature environment (0-70°C) with a single 5 volt ±10% power supply. All signal input voltage level specifications are identical. The input load currents and the output leakage currents are also the same. The operating currents (I_{DD1}, I_{DD2}, I_{DD3}, I_{DD4}) of the 2164A are greater than the 2118 because of the increased density of the 2164A. One other parametric difference worth pointing out is the maximum capacitive load of the control lines on the 2164A. The maximum specification is 8 pF on the $\overline{RAS}$ and $\overline{CAS}$ lines, each respectively 1 pF greater than the 2118.

1.3.1.2 A.C. Characteristics

As mentioned above, the t_{RAC} (access time from $\overline{RAS}$) spec of the 2164A-15 is a perfect match to the 2118-15. Generally, the other A.C. timing specs of the 2164A meet or exceed those of the 2118. Both the read and write cycle times (t_{RC}) of the 2164A-15 are 60 ns less than the 2118. The read-modify-write cycle of the 2164A runs 130 ns faster than the 2118. All parameters in the write cycle (reference 2164A data sheet page 3) of the 2164A exceed those of the 2118-15, as well as those timings specific to the read and refresh cycles. Noteworthy are the t_{RWL} (write command to $\overline{RAS}$ lead time) and t_{CWL} (write command to $\overline{CAS}$ lead time) specifications of the 2164A. These are 60 ns less than those of the 2118, allowing more flexibility in timing generation of the write cycle. One other improvement is t_{PC} (page mode read or write cycle) which is 125 ns. This parameter allows, for the first time, a two-fold performance advantage for page mode called extended page mode. This is offered as an option to read or write an entire page (row) of data during a single $\overline{RAS}$ cycle. By providing a fast t_{PC} and long $\overline{RAS}$ pulse width (t_{RPM2}), the 2164A-15 S6493 permits high-speed transfers of large blocks of data, such as required in bit-mapped graphics applications.

There are a few of the 2164A timing specifications however, that exceed those of the 2118. These are:

t_{CAC} (access from $\overline{CAS}$) = 85 ns, 5 ns greater than 2118

t_{RAH} (row address hold time) = 20 ns, 5 ns greater than 2118

t_{CAH} (col address hold time) = 25 ns, 5 ns greater than 2118

t_{RCD} ($\overline{RAS}$ to $\overline{CAS}$ delay time) = 30 to 65 ns, versus the 2118, 25 to 70 ns

Usually only the t_{RAH} specification has significance in system applications. This and all other system level compatibility issues are discussed in the following section.

1.3.2 SYSTEM LEVEL COMPATIBILITY

When designing a new system, the current (I_{DD}) requirements of the 2164A do not present any particular problems. Simply proceed with the normal power requirement analysis, and specify the power supply accordingly. (A method for determining memory system power requirements is detailed in Intel application note AP-131 titled: *Intel 2164A 64K Dynamic RAM Device Description.*) In a system being upgraded with 2164A devices, check the new power supply reqirements against the current power supply specifications to insure compatibility. Worth pointing out is the fact that in a 2118 system arranged as 64K by 16-bit word (32 devices) the power/bit of the 2118-15 is 2.6 microwatts/bit (see AP-75, pp. 11-12). Replacing the 2118's with 2164A DRAMS creates a 256K by 16-bit word (again, 32 devices) and the power per bit is 1.33 microwatts/bit (see AP-131, pp. 11-12). The quadrupling in memory size does not quadruple power supply requirements.

For a 64K by 16-bit to 256K by 16-bit conversion, the additional power required is 2.89 watts. (5.59 watts for the 2164A system — 2.7 watts for the 2118 system). On

the other hand, to build a 64K by 16-bit system with 2118 requires 2.7 watts versus only 1.4 watts for the 2164A, meaning that for a given system size, there is a significant system power system savings by implementing the design with the 2164A.

The difference in current (I_{DD}) specifications leads to another system consideration, that of decoupling. The larger current transients generated as a dynamic RAM internally powers up as a response to refresh cycles or active cycles requires decoupling to keep noise off the power grid and to prevent a transitory local voltage drop across devices. Specifics of calculating local and bulk decoupling requirements are presented in Section 6.3.4, but in general Intel recommends .1 μF high frequency ceramic capacitors for every 2164A device, and 100 μF bulk decoupling for every 32 devices.

In comparison to the 2118, the $\overline{RAS}$, $\overline{CAS}$ lines of the 2164A RAM have 1 pF additional load. This seems trivial on a device level, but in a system the extra capacitance adds approximately .1 ns/pF propagation delay (assuming low power Schottky drivers) to the overall system access path. With 16 devices per driver, this extra load adds up to a measurable increase in propagation delay. Determining additional delay due to capacitance is standard engineering practice in a new design. When upgrading a current memory system with 2164A DRAMs, the additional delay also has to be considered. Refer to section 6.2 for the formula to determine if the additional loading is a concern in any specific application.

Of the four timing specifications where the 2164A-15 exceeds the 2118-15 usually only t_{RAH} specification is of concern. If, however, the system being upgraded is $\overline{CAS}$ access limited rather than $\overline{RAS}$ access, then check the timing to determine if the extra 5 ns on t_{CAC} will require system re-tuning. The column address hold specification (t_{CAH}) needs also be checked in this case. In the majority of DRAM systems, the access speed of importance is t_{RAC}, the $\overline{RAS}$ access time. When optimizing a memory system to achieve the design's fastest access time, set the t_{RCD} spec to a value less than t_{RCD} maximum. In these high performance systems, be sure that the tighter 5 ns in the 2164A t_{RCD} spec window doesn't push out the system access time by that amount, or if it does, that it still conforms to the system timing requirements.

Reliablity qualification data for the 2164A and 2118 are identical with projections of less than .1%/1K-hrs for soft errors caused by α particles and less than .02%/1K-hrs for hard failures. This leads to a distinct system reliability advantage of the 2164A over the 2118. System reliability is qualified as MTBF (mean time between failure). This is the "up-time" of the system and is defined as $1/n\lambda$ where n is the number of devices in the system, and λ is the device failure rate. This equation (MTBF = $1/n\lambda$) says that system reliability is inversely proportional to the number of devices in the system. Therefore, a 1 Megabyte system (or any given system size) built with 2164A devices is four times more reliable as one built with 2118s.

In summary, when upgrading a system to 64K devices, increase the decoupling, check the power supply, and tweak the timing only if necessary, then enjoy the improved system reliability. When engineering a new design, become familiar with and be aware of the specification differences between the 2118 and the 2164A.

2 MICROPROCESSOR SYSTEM

To effectively design a microcomputer memory, an understanding of both the RAM and the microprocessor is necessary. Since Intel microprocessors have been well-documented in other publications, this applications note will mainly focus upon operation during bus cycles as related to the memory interface.

2.1 iAPX 86 Bus Operation

The iAPX 86 bus is divided into two parts: control bus and time-multiplexed address data bus. The bus is the microprocessor's only avenue for dialog with the system. The processor communicates with both the memory and I/O via the bus. As a result, it must necessarily differentiate between a memory cycle and an I/O cycle. In the minimum mode, this differentation is accomplished with the signal M/$\overline{IO}$ which remains valid during the entire cycle. Therefore, this signal need not be latched. In the maximum mode, the processor commits to a bus cycle by means of three status bits transmitted to the bus controller which generates the control signals.

The bus cycle is divided into four times, referred to as t-states, independent of the mode. Duration of this t-time (t_{CLCL}) is the reciprocal of the clock frequency into the microprocessor. During each of these states, a distinct suboperation occurs. In t_1, the address becomes valid and the system is informed of the type of bus cycle, memory or I/O. In addition, a clock called ALE (Address Latch Enable) is generated to enable the system to latch the address. This is required because the address will disappear in anticipation of data on the bus. Intended to strobe a flow-through latch, ALE becomes active after the address is valid and deactivated prior to the address becoming invalid. At the end of t_2, the Ready input is sampled. If it is low, the processor will "idle," repeating the t_3 state until the Ready line is high, allowing the memory or I/O to synchronize with the microprocessor. In t_3, the read or write operation commences and the high order status bits become valid.

Finally in t_4, the machine cycle is terminated; input data is latched into the processor in a read cycle or in a write cycle output data disappears. The relationship of the signals for minimum and maximum modes are shown in Figures 4 and 5. Exact timing relationships will be developed throughout the text. The design problem involves making the microprocessor signals intelligible to the dynamic RAM.

The timing analysis is to be given with a read cycle for the minimum mode configuration of Figure 6. Unlike static RAMs which access from whenever every input signal is stable, dynamic RAMs begin a cycle on a clock edge after addresses are stable. This will introduce a certain amount of delay in the logic path. The exact amount depends on the complexity of the memory controller. Two paths to access will be considered: first, the control signal to data input and second, address stable to data input. In the read cycle there are four control signals; M/$\overline{IO}$ used to differentiate memory and I/O cycles, $\overline{RD}$ used to control the output enable, DT/$\overline{R}$ and $\overline{DEN}$ are controls for data flow. Of these only M/$\overline{IO}$ is a concern to the memory design. Without WAIT states, no cycle can be longer than four clock periods.

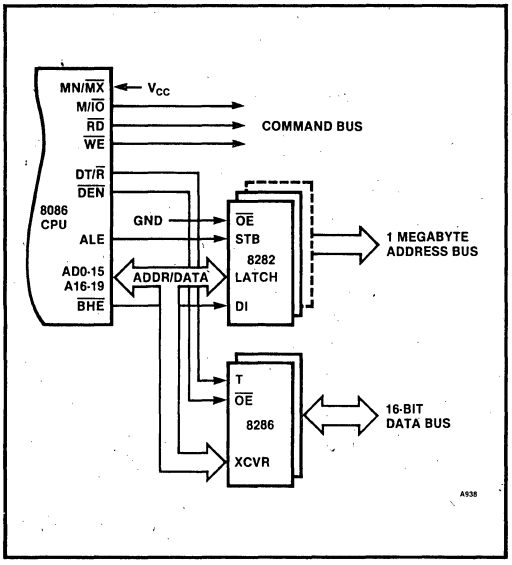


Figure 6. Minimum Mode Operation

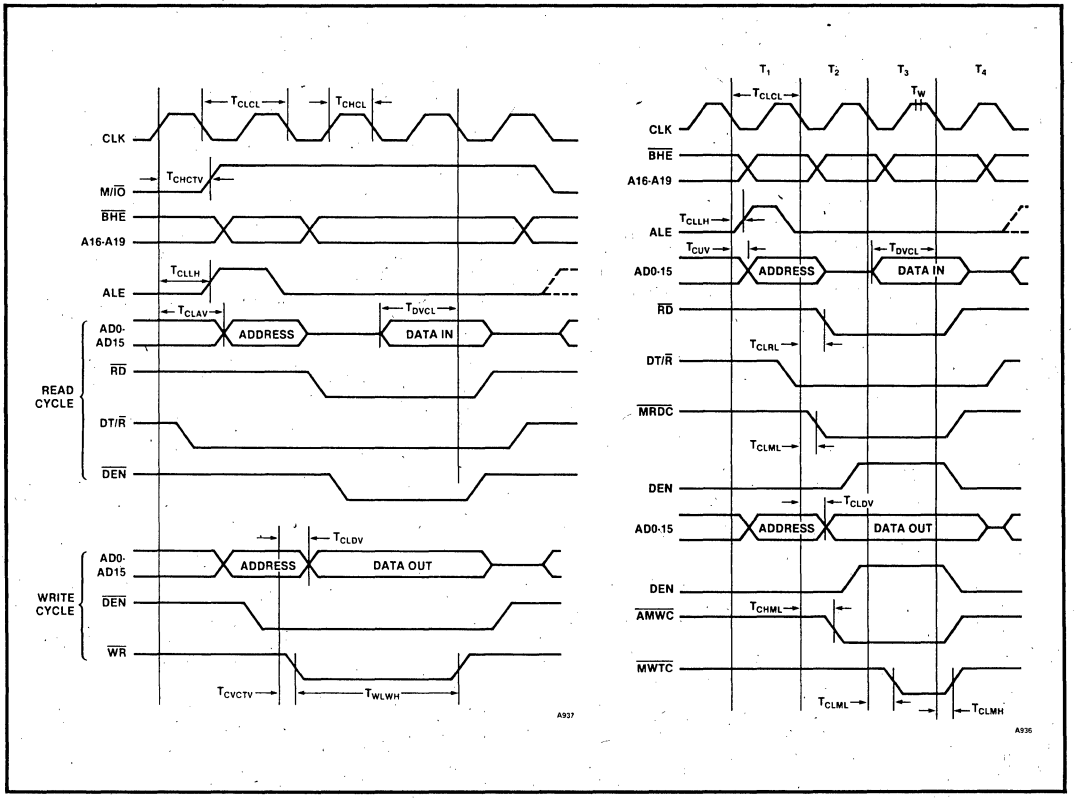


Figure 4. 8086 Bus Timing — Minimum Mode and Figure 5. 8086 Bus Timing — Maximum Mode

Referring to Figure 5, the following is obtained:

$MEMCY \leq 4\, t_{CLCL}$

M/$\overline{IO}$ is stable t_{CHCTV} from the previous clock high time t_{CHCL}, but;

$t_{CHCL} = 1/3t_{CLCL} + 2$

For the 5 MHz clock, $t_{CLCL} = 200$ ns

solving for t_{CHCL},
$t_{CHCL} = 68$ ns
But t_{CHCTV} is 110 ns.

As a result, M/$\overline{IO}$ is a stable worst case 32 ns after the start of a memory cycle. For an 8086-2, t_{CLCL} is 125 ns and t_{CHCTV} is 60 ns. Similarly, M/$\overline{IO}$ is stable 17 ns after the start of the cycle.

Address calculations must include the buffer delay (Figure 7). Stable addresses from the processor are available t_{CLAV} into the cycle and ALE is active t_{CLLH} into the cycle (Table 1). Addresses are on the bus t_{CLAV} plus t_{IVOV} (latch delay) or t_{CLLH} plus t_{SHOV} (buffer delay from strobe). The worst case number (t_{ADDR}) is the greater of these two numbers.

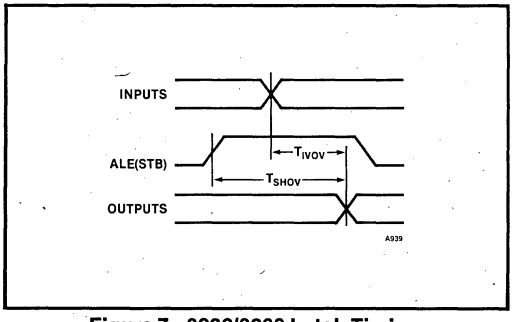


Figure 7. 8282/8283 Latch Timing

Table 1. Address Latch Delays — Min Mode

	5 MHz		8 MHz		10 MHz	
t_{CLAV} (ns) =	110		60		50	
t_{IVOV} (ns) =	+ 22		+ 22		+ 22	
Flow Thru (ns) =	132		82		72	
t_{CLLH} (ns) =		80		50		40
t_{SHOV} (ns) =		+ 40		+ 40		+ 40
Latch Delay (ns) =		120		90		80

Flow through delay is the limiting factor of the 5 MHz system, whereas delay from the latch strobe (ALE) is the limiting factor in the fastest processors. Finally, data must be inputted t_{DVCL} plus t_{IVOV} to the data buffer prior to the fourth t-state. Access from stable addresses is:

$$t_{ACC} = 3t_{CLCL} - t_{ADDR} - (t_{DVCL} + t_{IVOV})$$

Using this equation and the results from Table 1, t_{ACC} can be calculated.

Table 2. t_{ACC} Calculations — Min Mode

	5 MHz		8 MHz		10 MHz	
$3t_{CLCL}$ (ns) =		600		375		300
t_{ADDR} (ns) =	132		90		80	
t_{DVCL} (ns) =	+ 30		+ 20		5	
t_{IVOV} (ns) =	+ 22		+ 22		+ 22	
SUBTOTAL (ns) =	184	− 184	132	− 132	107	− 107
t_{ACC} (ns) =		416		243		193

Table 3 shows the system access time from stable address to input data required. This time is the summation of the RAM access time plus the control logic delay time.

Table 3. Data Setup Time — Min Mode

	5 MHz	8 MHz	10 MHz
t_{CVCTV} (ns) =	110	70	50
t_{CLDV} (ns) =	− 110	− 70	− 50
t_{IVOV} (ns) =	− 22	− 22	− 22
t_{DS} (ns) =	− 22	− 22	− 22

During a write cycle, access is not the issue, but the write pulse width, the data setup and hold time with respect to the write pulse are of concern. The pulse width is simply t_{WLWH}, while data set-up time must be calculated from a clock edge. Dynamic RAMs latch input data on the falling edge of the write enable pulse, so the calculation is critical. Data is valid t_{CLDV} plus the buffer delay t_{IVOV} in t_2 while the write pulse begins t_{CVCTV} in t_2. Worst case condition is a skew such that t_{CLDV} is a maximum delay while t_{CVCTV} has a minimum delay.

$$t_{DS} = t_{CVCTC} - (t_{CLDV} + t_{IVOV})$$

From the calculations in Table 3, the leading edge of the write pulse must be delayed in the minimum mode. These calculations will be used later.

Having examined the major timing parameters of the minimum mode configuration, let's now check the maximum mode timings.

In the maximum mode configuration of Figure 8, the system has another component — an 8288 bus controller — which generates ALE and the read and write control signals. In this configuration a memory read cycle is not committed until t_{CLML} into t_2 whereas in the minimum mode operation, the information was known in t_1. In this respect, a maximum mode system access cycle is less than $3t_{CLCL}$.

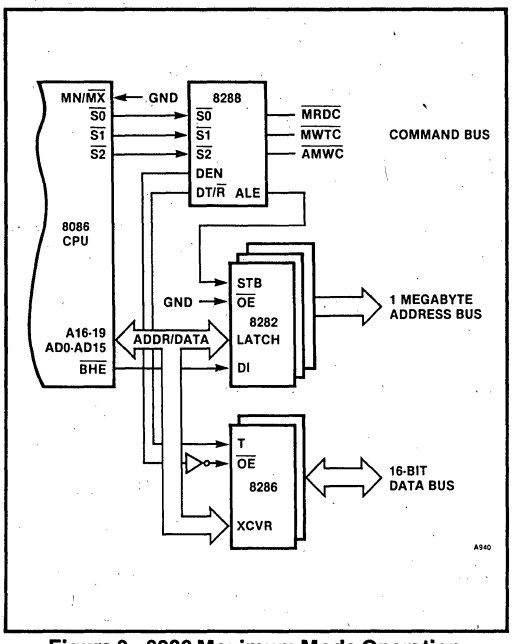


Figure 8. 8086 Maximum Mode Operation

To determine address delay, we will, again, examine the data flow path and the delay from the latch opening. The greater of these two numbers is the worst case time delay (t_{ADDR}).

$$\text{Flow-thru Delay} = t_{CLAV} + t_{IVOV}$$
$$\text{Latch Delay} = t_{CLLH} + t_{SHOV}$$

Using these equations and previous data, Table 4 shows how Flow-thru Delay can be calculated.

Table 4. Flow-through Delay — Max Mode

	5 MHz		8 MHz		10 MHz	
t_{CLAV} (ns) =		110		60		50
t_{IVOV} (ns) =		+ 22		+ 22		+ 22
Flow Thru Delay =		132		82		72
t_{CLLH} (ns) =	15		15		15	
t_{SHOV} (ns) =	+ 40		+ 40		+ 40	
Delay from ALE =	55		55		55	

In each case in Table 4, the limiting delay is flow-thru-time. Access time from address can now be determined. Again, data must be valid t_{DVCL} plus the input buffer delay (t_{IVOV}) before the end of t_3. For maximum mode access from the address valid time is:

$$t_{ACC} = 3t_{CLCL} - t_{ADDR} - (t_{DVCL} + t_{IVOV})$$

Using this equation and previous data (Table 4), Table 5 shows how t_{ACC} can be calculated.

Table 5. t_{ACC} Calculations — Max Mode

	5 MHz		8 MHz		10 MHz	
$3t_{CLCL}$ (ns) =		600		375		300
t_{ADDR} (ns) =	132		82		72	
t_{DVCL} (ns) =	+ 30		+ 20		5	
t_{IVOV} (ns) =	+ 22		+ 22		+ 22	
SUBTOTAL (ns) =	184	− 184	124	− 124	99	− 99
t_{ACC} (ns) =		416		251		201

Access from the read command ($\overline{MRDC}$) must also be determined. $\overline{MRDC}$ is valid t_{CLML} from t_2, causing access (t_{CA}) from $\overline{MRDC}$ to be:

$$t_{CA} = 2t_{CLCL} - t_{CLML} - (t_{DVCL} + t_{IVOV})$$

Using this equation, Table 6 shows the access calculations.

Table 6. Access From Memory Read Command

	5 MHz		8 MHz		10 MHz	
$2t_{CLCL}$ (ns) =		400		250		200
t_{CLML} (ns) =	35		35		35	
t_{DVCL} (ns) =	+ 30		+ 20		5	
t_{IVOV} (ns) =	+ 22		+ 22		+ 22	
SUBTOTAL (ns) =	87	− 87	77	− 77	62	− 62
t_{CA} (ns) =		313		173		138

Access from the memory read command ($\overline{MRDC}$) is much more stringent than address access. Consequently both access paths must be consideed in system design. The write cycle has the same limitation as access from memory read command. Memory write is identified by $\overline{MWTC}$ having the same timing as the memory read command. Address timing is the same for both the read and write cycles. The write pulse, t_{WP} is generated by $\overline{MWTC}$ with a pulse width of one clock cycle plus maximum t_{CLML} plus the minimum overlap into the next cycle (t_{CLMH}).

$$t_{WP} = t_{CLCL} + t_{CLML} - t_{CLMH}$$

For the 5 MHz, 8 MHz, and 10 MHz system, t_{WP} is calculated as shown in Table 7.

Table 7. t_{WP} Calculations — Max Mode

	5 MHz		8 MHz		10 MHz	
t_{CLCL} (ns) =		200		125		100
t_{CLML} (ns) =	35		35		35	
t_{CLMH} (ns) =	− 10		− 10		− 10	
SUBTOTAL (ns) =	25	− 25	25	− 25	25	− 25
t_{WP} (ns) =		175		100		75

Data setup time (t_{DS}) to the leading edge of the write pulse occurs approximately one t_{CLCL} time later. From t_{CLCL}, the maximum t_{DVCL} plus the minimum t_{CLML} must be subtracted:

$$t_{DS} = t_{CLCL} - (t_{CLDV} + t_{CLML})$$

Now t_{DS} can be computed as shown in Table 8 by using data from previous calculations and the data sheet.

Table 8. Data Setup (t_{DS}) Calculations — Max Mode

		5 MHz	8 MHz	10 MHz
t_{CLCL} (ns)	=	200	125	100
t_{CLDV} (ns)	=	−110	− 60	− 50
t_{CLML} (ns)	=	− 10	− 10	− 10
t_{DS} (ns)	=	80	55	40

Using $\overline{MWTC}$ as the write pulse allows sufficient data set-up time for the dynamic RAMs. These, then, are the basic timing equations for the system of Figures 6 and 8. They are general in that timing requirements for different clock frequencies (i.e., 9 MHz) can be calculated using them. Armed with these equations, the designer can now shape the control and address signal in the time domain with a memory controller to meet the dynamic RAM requirements.

In addition to converting address, $\overline{MRDC}$ and $\overline{MWTC}$ into $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, etc., to satisfy both the processor and memory, another task called refresh must be performed by the memory controller.

Performing the interface translation, providing refresh and controlling the signal timing to the RAM requires a controller that consists of six elements as shown in Figure 9. Of these, the most basic is the oscillator because it fulfills two functions: providing a time base for refresh interval timing and establishing precise times for $\overline{RAS}$, $\overline{CAS}$, etc., to the RAM. The operating frequency must be high enough to provide sufficient increments between timing signals. The relationship of timing signals will be multiple periods of the clock frequency. In addition, the oscillator drives a countdown or divide by N circuit to measure the time between refresh cycles. Refresh can be either burst or distributed. In the burst mode, a refresh request would occur once every two milliseconds to meet the dynamic RAMs' needs. For a 16K or 64K RAM with 128 refresh cycle requirement, all 128 refresh cycles would be performed consecutively. A disadvantage of this method is that the memory is "out of service" for a long period of time. Assume a 350 ns cycle time, then the time required to perform refresh is 350 ns multiplied by 128 cycles or 44.8 microseconds operating with a 5 MHz 8086; this translates to 224 consecutive WAIT states.

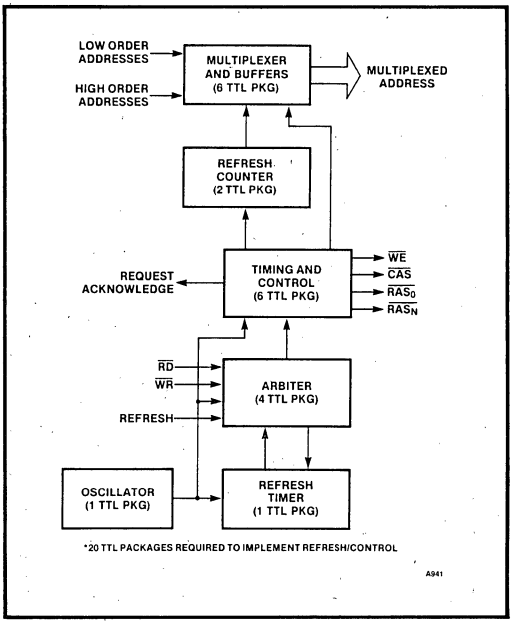


Figure 9. Refresh Timing and Control Block Diagram

Consequently, a large delay is injected every 2 ms. On the other hand, distributed refresh steals a single cycle, 128 times periodically throughout the 2 ms. Evenly distributed, a refresh cycle occurs once every 15 microseconds. Again assume a 350 ns refresh cycle, and our 5 MHz system need only inject two WAIT states (worst case) each time. Thus distributed refresh is preferable in almost all microprocessor systems.

Guaranteeing that all 128 refresh addresses are exercised is the task of the refresh address counter. It consists of an eight-stage binary counter. After the refresh cycle has been completed, the counter is advanced one count. Incrementing after refresh eliminates any concern regarding address settling or setup time as the counter outputs are changing. This would be a concern if the counter were incremented as the refresh cycle started.

Because the counter cycles through all 128 addresses every 2 milliseconds, it isn't required to be in a specific state after power on, i.e., it need not start at address 0 after power on.

Next is the arbiter — which can be the bane of every memory design. Deciding whether a memory cycle is an access cycle or a refresh cycle is the function of the arbiter. Refresh requests are derived from the oscillator which operates asynchronously with the system clock. The arbiter will grant the request when a refresh request is made and no memory cycle is occuring or pending. If

an access cycle is in progress, the arbiter must inhibit the refresh cycle until the current cycle is completed. The same logic process occurs if a refresh cycle is in progress and access is requested. This sequence flows smoothly most of the time. The difficulty arises when refresh and access are requested simultaneously. In every arbiter there exists an infinitely small but very real time period when the arbiter cannot make a decision, much less the correct one. Consider the arbiter in Figure 10 — a simple cross-coupled NAND or an R-S flip-flop.

If both requests are made simultaneously, both would be granted — an impossibility!

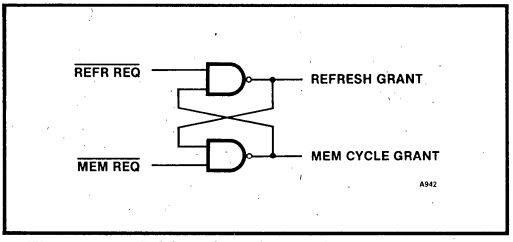


Figure 10. Arbiter Cross-coupled NAND Gates

Another arbiter frequently used is a D-type flip-flop as in Figure 11. Here arbitration is attempted between the clock and the D input. Violating the setup or hold time with respect to the clock can cause the output to enter a quasi-stable state of non-TTL levels for as long as 75 ns. This timing is too long for many high performance systems.

Effective solutions have reduced performance to maximize reliability. One such method is a two stage clocked flip-flop per Figure 12.

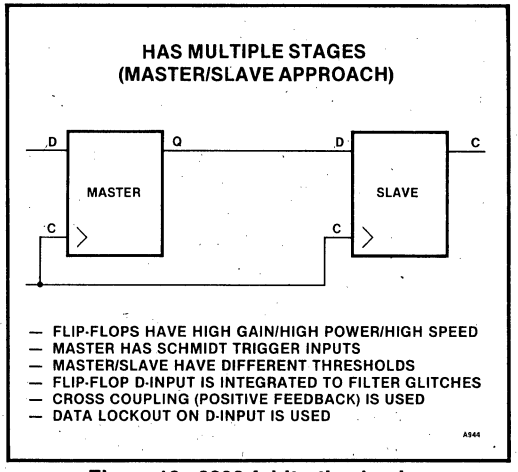


Figure 12. 8203 Arbitration Logic

In this configuration arbitration is performed at the second stage so that even if the first stage "hangs" all will be settled by the clocking of the second stage.

The timing and control section is the core of the controller. Under its guidance, addresses are switched for multiplexing. $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ are produced and sequenced in a fashion understandable by the RAMs. One other fea-

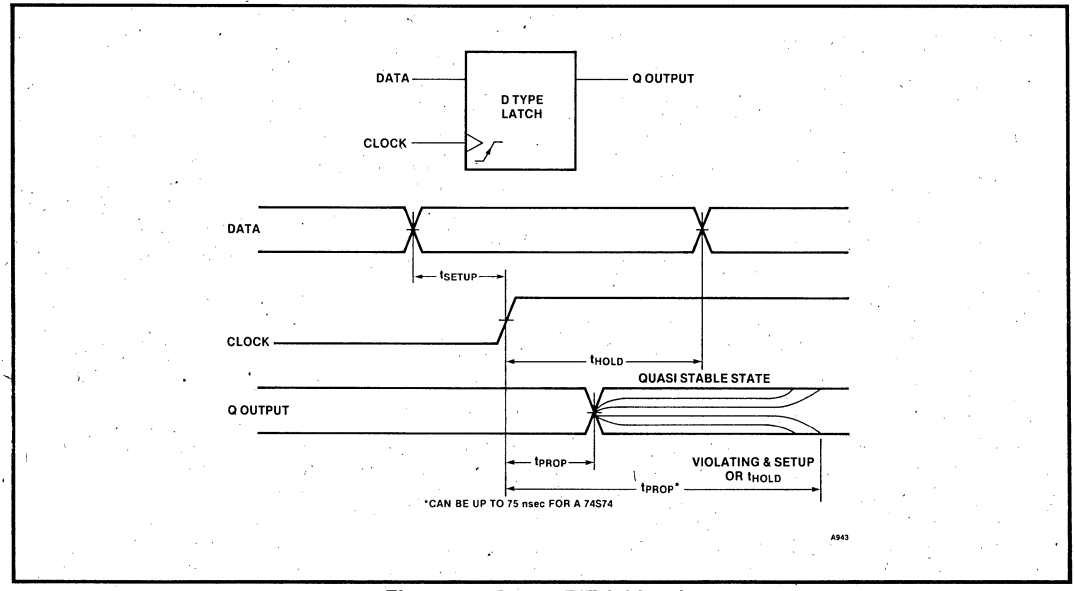


Figure 11. D-type F/F Arbitration

ture required is a handshake signal with the processor to indicate whether or not the memory is ready to be accessed. This is usually implemented with a System Acknowledge ($\overline{SACK}$) (an early signal in the cycle) which indicates a receipt by the controller of a memory access request, or by a Transfer Acknowledge ($\overline{XACK}$, a signal occuring later in the memory cycle), indicating the valid memory data is available.

The final piece of the memory controller is the address multiplexers and buffers to drive the memory addresses. During the normal memory cycle the parallel addresses from the bus must be reduced by one half through time multiplexing. In addition refresh addresses must be applied to the array through this same address path. Buffers are shown to drive the capacitance of the array with signals having sharp rise and fall times.

Figure 9 also shows the quantity of TTL packages required to implement such a controller. Twenty TTL packages are usually required for a controller.

To design a controller with discrete TTL components can take several man months of design effort. Typically, four weeks for design, two weeks for timing analysis, four weeks to build and debug prototypes, six weeks for circuit board layout, and another four weeks to add additional features or to tweak the original design. Obviously, the Intel 8203 DRAM controller is a desirable alternative.

2.2 8203 Dynamic RAM Memory Controller

The Intel 8203 is a Schottky bipolar device housed in a 40-pin dual in-line package. It provides a complete dynamic RAM controller for microprocessor systems and expansion memories. All of the system control signals are provided to operate and refresh the 2117, 2118 and 2164A dynamic RAMs. To accomplish this, the 8203 provides the following features:

- Directly addresses one-half megabyte of 2164A (with external drivers)
- Provides address multipexing and $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ strobes
- Provides a refresh timer and an 8-bit refresh address counter
- Refresh may be internally selected for automatic refresh in a distributed fashion
- Refresh may be externally requested to provide for synchronous or transparent refresh
- Compatible with Intel 8080A, 8085A, iAPX 88 and iAPX 86 families of microprocessors
- Provides system acknowledge and transfer acknowledge signals
- Allows asynchronous memory and refresh cycle requests
- Provisions for external clock or crystal oscillator

A block diagram of the 8203 is given in Figure 13 which illustrates how these features are integrated.

2.2.1 OSCILLATOR

The Intel 8203 generates its timing from an internal shift register which is crystal controlled. This method provides highly accurate control of the timing required for dynamic RAMs. This method is superior to a monostable mulitvibrator approach where transients and unit-to-unit timing accuracies are difficult to control.

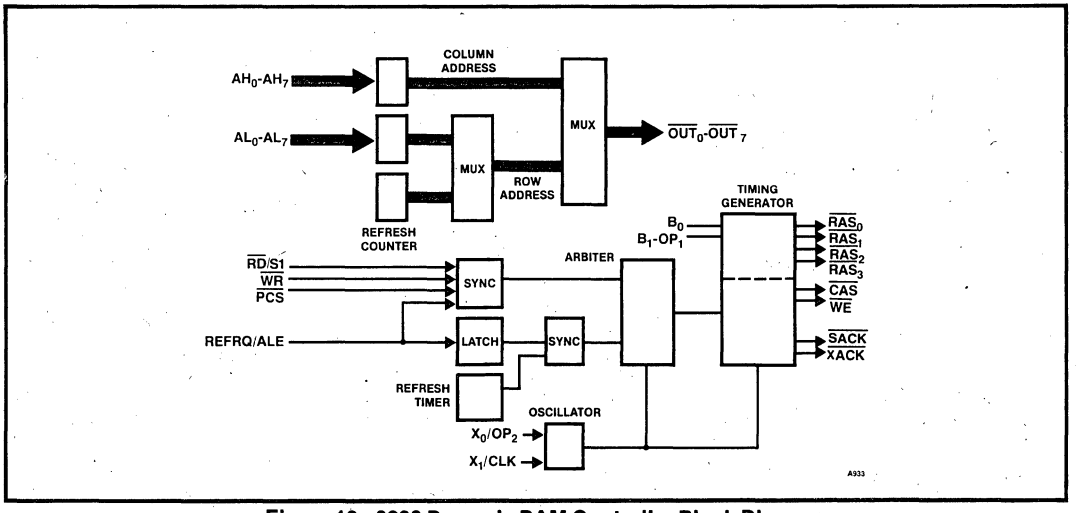


Figure 13. 8203 Dynamic RAM Controller Block Diagram

2.2.2 ARBITER

The arbiter resolves all conflicts between any cycles that are requested simultaneously. These cycles can be generated from one of four places:

1. Read Cycle Request — $\overline{RD}$/S1 input
2. Write Cycle Request — $\overline{WR}$ input
3. External Refresh Request — REFRQ/ALE
4. Internal Refresh Request — (refresh timer shown in Figure 13)

If a refresh cycle is in progress and a read or write cycle is requested, the requesting device receives a "not ready" until the present cycle is completed. After completion of the present refresh cycle a response from the 8203 called System Acknowledge, or $\overline{SACK}$, will notify the requesting device of availability for use. If a read or write request occurs simultaneously with a refresh request, the read or write cycle will be performed first, then the refresh cycle. Read and write cycle requests cannot occur simultaneously during normal operation. If the 8203 is deselected, only an internal or external refresh cycle request will be accepted. Once selected, it will continue with the present memory cycle if one is being performed. (Hence the chip select input is called protected chip select, $\overline{PCS}$, because the current cycle is always completed regardless of any other pending request.)

2.2.3 REFRESH TIMER AND COUNTER

The refresh timer is a counter that increments on each pulse from the clock input until it reaches a preset number causing an internal refresh request to occur. Note that this causes the refresh rate to be 8203 clock cycle dependent. External refresh requests will cause the refresh timer to reset, but will not disable it.

The internal address counter contains the address that will be used during the next refresh cycle. The counter is incremented after each refresh, counting up to 256 before resetting to zero after all RAM addresses have been refreshed. All current generation Intel DRAMs require a 128-cycle refresh, hence, the most significant bit is ignored. However, this extra bit allows use of 256 cycle 4 ms refresh devices without changing the current memory system design.

2.2.4 MULTIPLEXER

The multiplexer is controlled by the timing and control logic. It presents to the address bus one of the following:

1. The contents of the refresh counter when there is a refresh cycle
2. $AL_{0\text{-}6}$ on a $\overline{RAS}$ pulse
3. $AH_{0\text{-}6}$ on a $\overline{CAS}$ pulse

The outputs from the multiplexer are inverted from the address inputs. This is immaterial to the dynamic RAM array and does not require inversion for proper system operation.

2.2.5 TIMING AND CONTROL

The timing and control logic allows either a read, write or refresh cycle to occur. After any read or write cycle request, $\overline{SACK}$ (System ACKnowledge) goes active if the cycle was not requested during a refresh cycle. If it was, $\overline{SACK}$ is delayed until $\overline{XACK}$, thereby requesting WAIT states from the cycle requester.

Figure 14 is a diagram of the 8203 pinout. Table 9 lists the pin numbers, the symbols, and the function of each pin when the 8203 is configured for the 64K option.

The 8203 has two ways of providing dynamic RAM refresh:

1. Internal (failsafe) refresh
2. External refresh

Both types of 8203 refresh cycles activate all of the $\overline{RAS}$ outputs, while $\overline{CAS}$, $\overline{WE}$, $\overline{SACK}$, and $\overline{XACK}$ remain inactive.

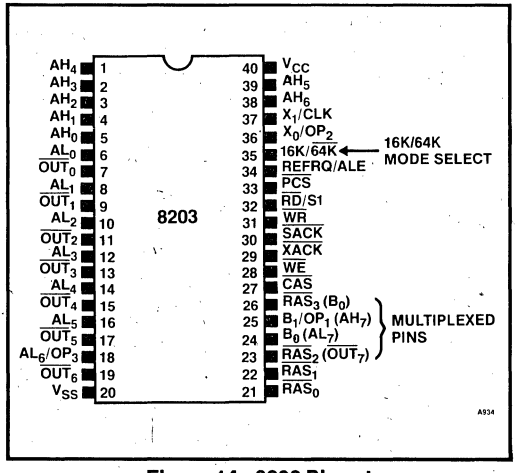


Figure 14. 8203 Pinout

2.2.6 REFRESH CYCLES

Internal refresh is generated by the on-chip refresh timer. The timer uses the 8203 clock to ensure that refresh of all rows of the dynamic RAM occurs every 2 milliseconds. If REFRQ is inactive, the refresh timer will request a refresh cycle every 10-16 microseconds.

External refresh is requested via the REFRQ input (pin 34). External refresh control is not available when the Advanced-Read mode is selected. External refresh requests are latched, then synchronized to the 8203 clock.

Table 9. Pin Description (64K Option)

Symbol	Pin No.	Type	Name and Function
AL_0	6	Input	Address Low: CPU address inputs used to generate memory row address.
AL_1	8	Input	
AL_2	10	Input	
AL_3	12	Input	
AL_4	14	Input	
AL_5	16	Input	
AL_6	18	Input	
AL_7	24	Input	
AH_0	5	Input	Address High: CPU address inputs used to generate memory column address.
AH_1	4	Input	
AH_2	3	Input	
AH_3	2	Input	
AH_4	1	Input	
AH_5	39	Input	
AH_6	38	Input	
AH_7	25	Input	
BO	26	Input	Bank Select Input: Used to gate the appropriate $\overline{RAS}_0$-$\overline{RAS}_1$ output for a memory cycle.
$\overline{PCS}$	33	Input	Protected Chip Select: Used to enable the memory read and write inputs. Once a cycle is started, it will not abort even if $\overline{PCS}$ goes inactive before cycle completion.
$\overline{WR}$	31	Input	Memory Write Request
$\overline{RD}$	32	Input	Memory Read Request
REFRQ	34	Input	External Refresh Request
$\overline{OUT}_0$	7	Output	Output of the Multiplexer: These outputs are designed to drive the addresses of the dynamic RAM array. (Note that the $\overline{OUT}_{0\text{-}7}$ pins do not require inverters or drivers for proper orientation.)
$\overline{OUT}_1$	9	Output	
$\overline{OUT}_2$	11	Output	
$\overline{OUT}_3$	13	Output	
$\overline{OUT}_4$	15	Output	
$\overline{OUT}_5$	17	Output	
$\overline{OUT}_6$	19	Output	
$\overline{OUT}_7$	23	Output	
$\overline{WE}$	28	Output	Write Enable: Drives the write enable inputs of the dynamic RAM array.
$\overline{CAS}$	27	Output	Column Address Strobe: This output is used to latch the column address into the dynamic RAM array.
$\overline{RAS}_0$	21	Output	Row Address Strobe: Used to latch the row address into bank of dynamic RAMs, selected by the 8203 Bank Select Pin (B_0).
$\overline{RAS}_1$	22	Output	
$\overline{XACK}$	29	Output	Transfer Acknowledge: This output is a strobe indicating valid data during a read cycle or data written during a write cycle. $\overline{XACK}$ can be used to latch valid data from the RAM array.
$\overline{SACK}$	30	Output	System Acknowledge: This output indicates the beginning of a memory access cycle. It can be used as an advanced transfer acknowledge to eliminate WAIT states. (Note: if a memory access request is made during a refresh cycle, $\overline{SACK}$ is delayed until $\overline{XACK}$ in the memory access cycle.)
X_0/OP_2	36	Input	Crystal Inputs: These inputs are designed for a quartz crystal to control the frequency of the oscillator. X_1/CLK becomes a TTL input for an external clock if X/OP is tied to V_{CC}.

The arbiter will allow the refresh request to start a refresh cycle only if the 8203 is not in a cycle.

Internally, if a memory request and a refresh request reach the arbiter at the same time, the 8203 will honor the refresh request first. However, the external refresh synchronization takes longer than the memory request synchronization so, relative to the 8203 input signals, a simultaneous memory request and external refresh request will result in the memory request being honored first. This 8203 characteristic can be used to "hide" refresh cycles during system operation. A circuit similiar to Figure 15 can be used to decode the CPU's instruction fetch status to generate an external refresh request. The refresh request is latched while the 8203 performs the instruction fetch: the refresh cycle will start immediately after the memory cycle is completed, even if the $\overline{RD}$ input has not gone inactive. If the CPU's instruction decode time is long enough, the 8203 can complete the refresh cycle before the next memory request is generated.

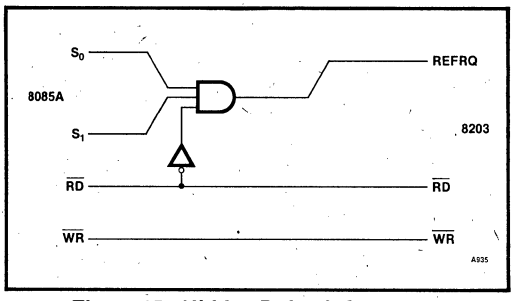


Figure 15. Hidden Refresh Generator

After each refresh cycle, the 8203 increments the refresh counter, reloads the refresh timer, and clears the external refresh latch. If the external refresh request is held active, the latch will be set again, and another refresh cycle will be generated. If, however, a memory request is pending, it will be honored before the second refresh request. This feature prevents refresh from locking out the memory request.

Certain system configurations require complete external refresh control. If external refresh is requested faster than the minimum internal refresh timer (t_{REF}) then, in effect, all refresh cycles will be caused by the external refresh request, and the internal refresh timer will never generate a refresh request.

2.2.7 READ CYCLES

The 8203 can accept two different types of memory Read requests:

1. Normal Read, via the $\overline{RD}$ input
2. Advanced Read, using the S1 and ALE inputs

The user can select the desired Read request configuration via the B1/OP1 hardware strapping option on pin 25.

Normal Reads are requested by activating the $\overline{RD}$ input, and keeping it active until the 8203 responds with an $\overline{XACK}$ pulse. The $\overline{RD}$ input can go inactive as soon as the command hold time (t_{CHS}) is met.

Advanced Read cycles are requested by pulsing ALE while S1 is active; if S1 is inactive (low) ALE is ignored. Advanced Read timing is similiar to Normal Read timing, except the falling edge of ALE is used as the cycle start reference.

If a read cycle is requested while a refresh cycle is in progress, then the 8203 will set the internal delayed-$\overline{SACK}$ latch. When the Read cycle is eventually started, the 8203 will delay the active $\overline{SACK}$ transition until $\overline{XACK}$ goes active. This delay was designed to compensate for the CPU's READY setup and hold times. The delayed-$\overline{SACK}$ latch is cleared after every READ cycle.

Based on system requirements, either $\overline{SACK}$ or $\overline{XACK}$ can be used to generate the CPU READY signal. $\overline{XACK}$ will normally be used; if the CPU can tolerate an advanced READY, then $\overline{SACK}$ can be used. If $\overline{SACK}$ arrives too early to provide the appropriate number of WAIT states, then either $\overline{XACK}$ or a delayed form of $\overline{SACK}$ should be used.

2.2.8 WRITE CYCLES

Write cycles are similiar to Normal Read cycles, except for the $\overline{WE}$ output. $\overline{WE}$ is held inactive for Read cycles, but goes active for Write cycles. All 8203 Write cycles are "early write" cycles; $\overline{WE}$ goes active before $\overline{CAS}$ goes active by an amount of time sufficient to keep the dynamic RAM output buffers turned off.

For a more detailed analysis of the 8203, refer to Application Note AP-97A, entitled "Interfacing Dynamic RAMs to iAPX 86/88 Systems Using the Intel 8202A and 8203."

3 SIMPLE SOLUTION

An example of the ease of interfacing DRAMs to microprocessors with the 8203 is shown in Figure 16. This is an example of the 8203 and 2118's or 2164A's configured as local memory to a min mode iAPX 88 System. The CPU's local bus is demultiplexed by an 8283 which latches the addresses and presents them to the 8203. Notice the lack of TTL support circuitry. The only additional components are a latch for the dynamic RAM output data and a OR gate to steer the $\overline{WE}$ signal on byte writes. The 8203 handles all the interface requirements of the

DRAM array, rendering a very simple solution to a dynamic memory design.

Figure 17 is an 8203/2164A memory system configured as a global resource to a max-mode iAPX 86 microprocessor system. Although there are several more TTL components involved, the buffers and transceivers are a requirement for proper system bus interface design. In terms of controlling the memory, the 8203 and 2164A interface is as simple as in the previous example. The ability of the 16 bit 8086 to perform byte operations requires two gates (shown on the diagram of Figure 17 between the 8203 and the 2164A array) to steer the write pulse output from the 8203 to either the high or low byte or both bytes as directed by A0 and $\overline{BHE}$ (Byte High Enable).

These examples balance ease of use and design throughput time with performance. The designs shown typically require one to two WAIT states. With one WAIT state,

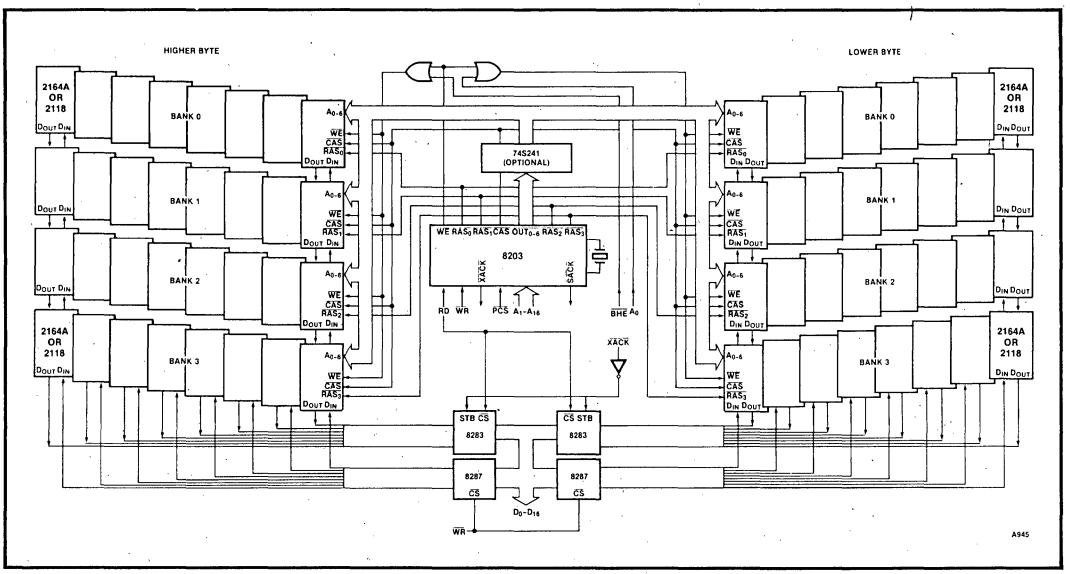


Figure 16. 8203/2118 Local Memory System

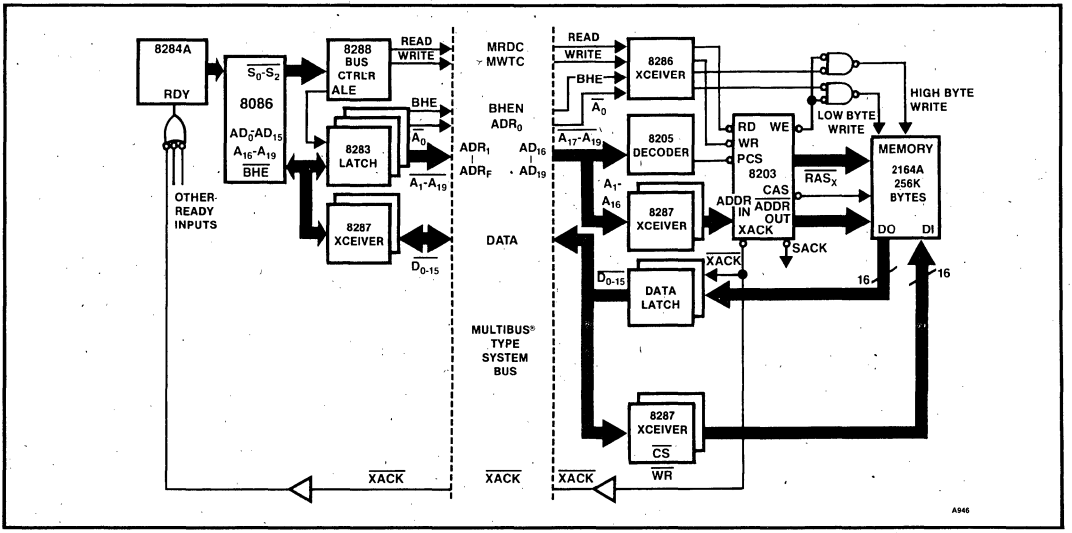


Figure 17. 8203/2164A Global Memory System

processor performance is reduced to 91.7%, and with two WAIT states it drops to 83.7%. This may be acceptable in many applications, but where it is not, a modest additional design effort can yield zero WAIT states.

4 5 MHz NO-WAIT STATE SYSTEM1

4.1 Circuit Description

The DRAM/8203 microprocessor memory system discussed up to this point met all of our design criteria except one — optimum performance. In minimum mode operation, inherent delays in the system $\overline{RD}$ and $\overline{WR}$ commands resulted in a READY signal that was too late to avoid processor WAIT states. Attaining zero WAIT states requires minimizing these delays by transmitting advanced read ($\overline{RD}$) or write ($\overline{WR}$) commands. This is not a simple task in minimum mode operation because the iAPX 86 processor produces the $\overline{RD}$ and $\overline{WR}$ signals in a fixed relationship after ALE occurs. However, operating in a max-mode, the iAPX 86 outputs three status bits ($\overline{S0}$, $\overline{S1}$, $\overline{S2}$) which occur ahead of the ALE signal. (Refer to the timing diagram shown in Figure 18.) With proper logic circuitry, these status bits can be used to initiate the advanced signals required.

The following discussion describes a 5 MHz no-WAIT state microprocessor memory system designed for optimum performance. Figure 19 shows an iAPX 86 maximum mode system modified for zero WAIT states. The circuitry added to the system previously described is enclosed in the dashed lines. The 8205 decodes the three status bits ($\overline{S0}$, $\overline{S1}$, $\overline{S2}$) and outputs an advanced read or write signal at pin 13 or 14, respectively. These signals flow through the corresponding 74S158 (a 2:1 mux con-

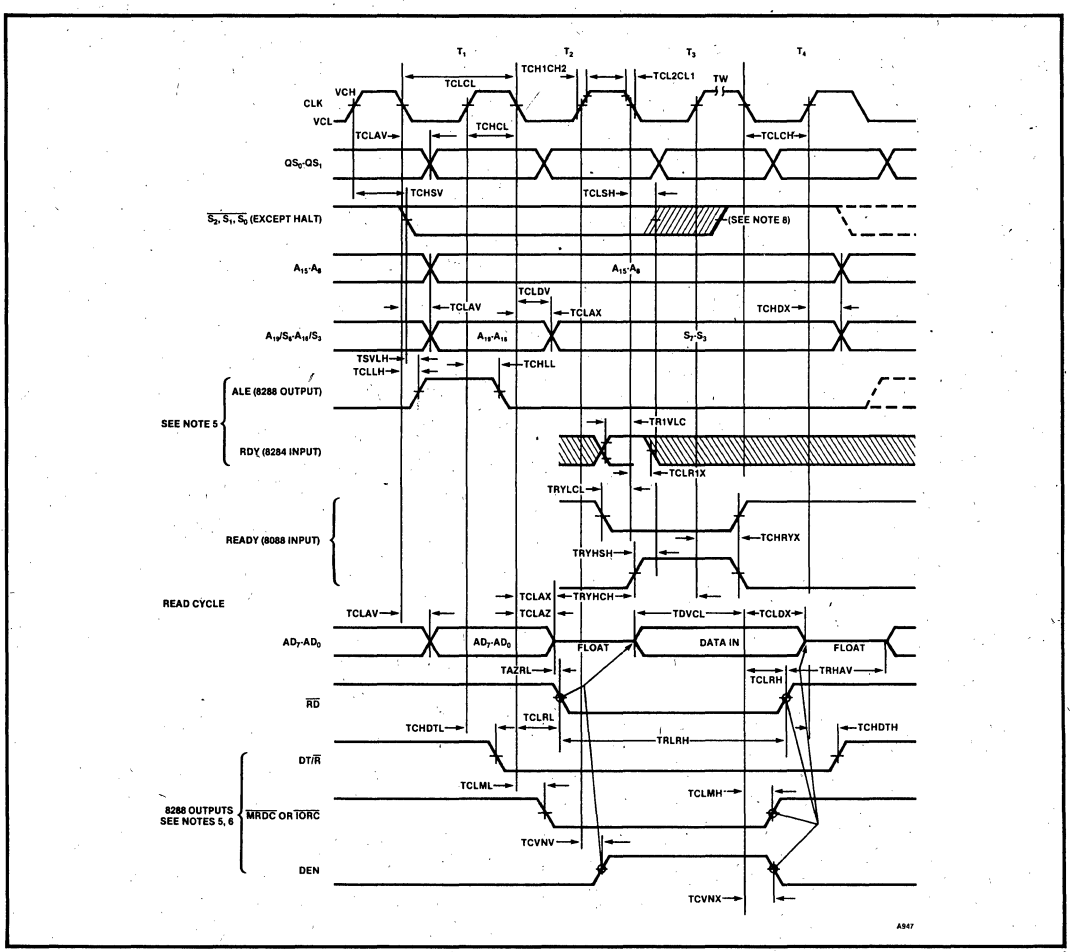


Figure 18. 8086 Bus Timing — Maximum Mode System (using 8288)

figured as a high speed flow through latch) and are latched on the falling edge of ALE from the 8288. Latch outputs (ADV $\overline{WRC}$ and ADV $\overline{RDC}$) are connected to the 8203 $\overline{WR}$ and $\overline{RD}$ inputs. The two latches are cleared by clocking the trailing edge of either the memory read command ($\overline{MRDC}$) or memory write command ($\overline{MWTC}$) through a 74S74 flip-flop. System acknowledge ($\overline{SACK}$ — used in place of $\overline{XACK}$ because it occurs sooner) is returned to the 8284A which provides a synchronous ready signal to the iAPX 86. The advanced memory write command, $\overline{AMWC}$, clocked to provide appropriate timing with $\overline{CAS}$, is ORed with $\overline{WE}$ to obtain the $\overline{WR}$ for the 2118's. The $\overline{S2}$ status bit is latched by the 74S158 on the trailing edge of ALE.

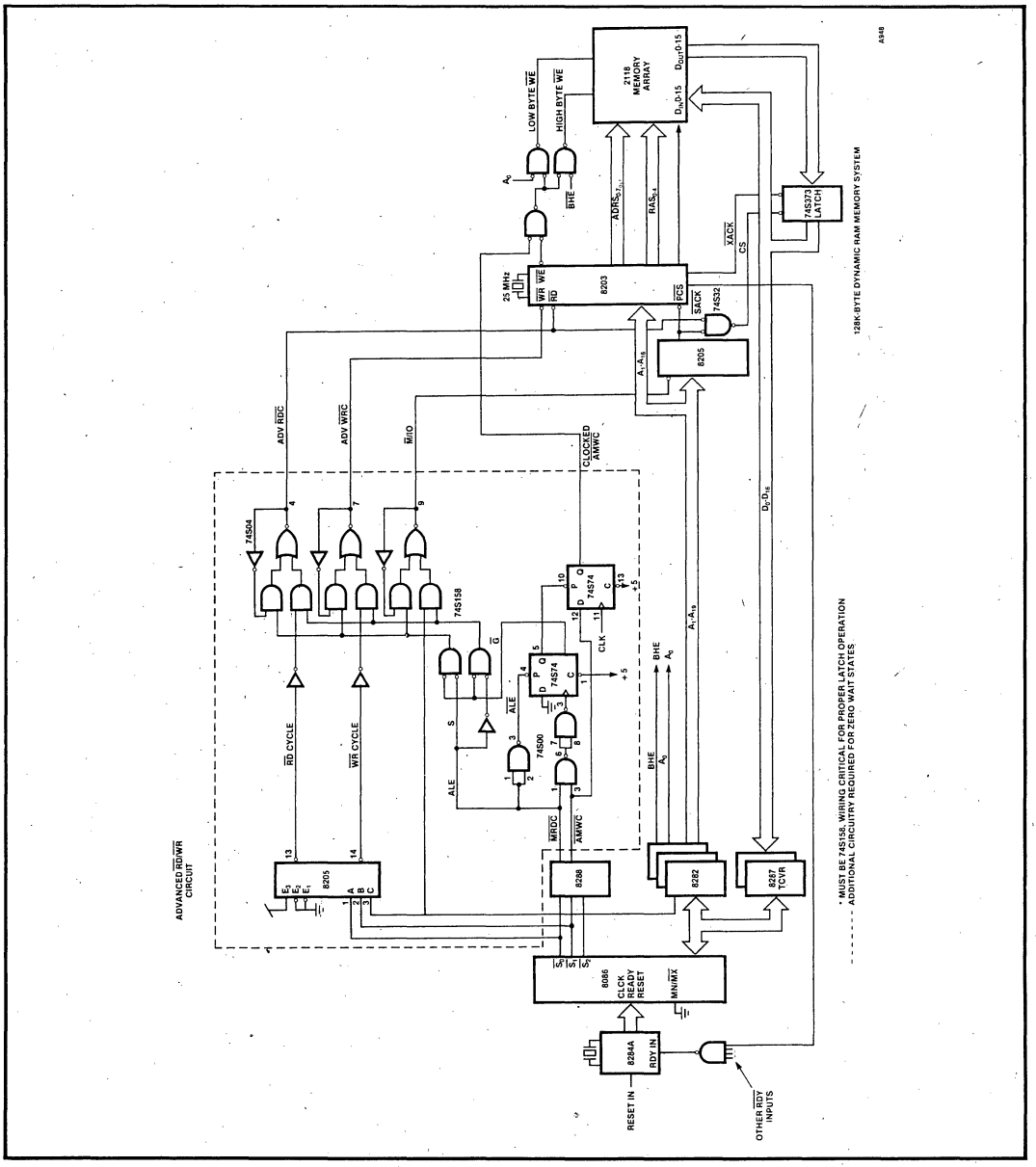


Figure 19. 5 MHz No-WAIT State Microprocessor Memory System

4.2 Analysis and Description of System Timing

Read cycle worst case analysis is shown in Figure 20 which only considers the maximum time delays. The four processor t states are indicated by t_1 through t_4. To accomplish zero WAIT states, valid data must reach the iAPX 86 by the end of t_3 minus 30 ns. The latest read data arrives at the iAPX 86 (next to the last waveform) within this time frame. Timing relationships are as follows:

The ADV $\overline{RDC}$ flows through the 74S158 latch and reaches the 8203 within 6 ns after the rise of ALE. The latest $\overline{PCS}$ is generated by decoding CPU addresses and arrives within 133 ns. The $\overline{SACK}$ signal is then returned within 127 ns from $\overline{PCS}$. The buffered $\overline{SACK}$ is used as the READY signal to the iAPX 86, resulting in zero WAIT states (except when the 8203 is performing a refresh cycle). The maximum $\overline{PCS}$ to $\overline{CAS}$ delay is shown to be 245 ns. Also accounted for is the maximum access time from $\overline{CAS}$ to data valid of 80 ns and a propagation delay of 45 ns for valid data to reach the processor.

In the write cycle, the relationship between data and $\overline{WE}$ at the memory and the relationship between the leading edge of $\overline{WE}$ and the trailing edge of $\overline{CAS}$ (t_{CWL}) must be

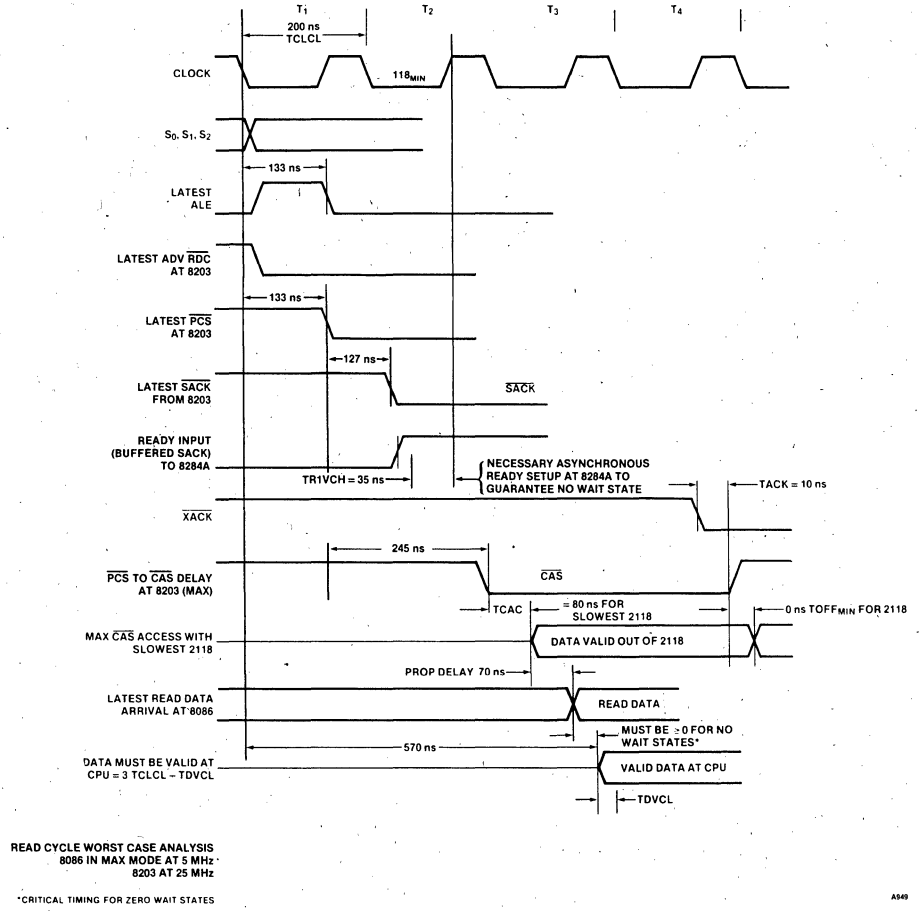


Figure 20. Read Cycle Timing Analysis (5 MHz)

preserved. Since DRAMs write data on the leading edge of the write pulse, data must be valid before the fall of $\overline{WE}$. Timing analysis of the skew of the normal memory write command ($\overline{MWTC}$) to valid data shows that worse case, it is possible to have data arrive after the falling edge of $\overline{WE}$ (case 1 of Figure 21). Using the other write pulse available from 8288 bus controller, the advanced memory write command ($\overline{AMWC}$), led to the problem depicted in Figure 21, case 2, violation of the DRAM specification t_{CWL}. From these observations, the need for the clocked $\overline{AMWC}$ pulse becomes apparent. By delaying the $\overline{AMWC}$ pulse until the next rising edge of the system clock and then gating this signal with the $\overline{WR}$ output from the 8203, a "best-fit" write pulse is created that meets all timing requirements.

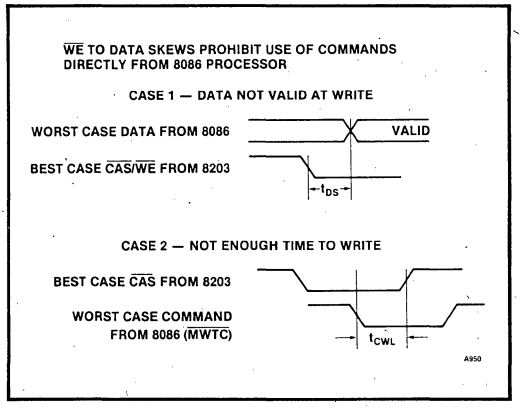


Figure 21. Write Cycle Problems

Figure 22 depicts the worst case analysis of the write cycle. The timing relationships are similar to those for the read cycle with a few exceptions. The advanced write command, ADV $\overline{WRC}$, flows through 74S158 and is latched by the fall of ALE. The earliest $\overline{CAS}$ occurs 145 ns after the $\overline{PCS}$. Valid data is output from the CPU within 210 ns and reaches the memory 35 ns later. The advanced memory write command, $\overline{AMWC}$, and associated progation delays must satisfy the t_{CWL} requirement of the 2118's which starts at the beginning of the $\overline{AMWC}$ pulse and terminates with the end of $\overline{CAS}$. The write enable, $\overline{WE}$, from the 8203, is ANDed with $\overline{AMWC}$ to obtain the $\overline{WR}$ for memory.

4.3 Compatibility of the 2118 and 2164A

The 5 MHz no-WAIT state system was designed with the 2118-15 DRAM. By following the guide lines in section 1.3 and examining tight timing areas specific to this application, it can be shown that the system is expandable and works equally well by using two rows of 2164A-15 parts in place of four rows of 2118-15 parts. The 8203, when configured in the 64K mode, guarantees proper generation and arrival of timing signals to the memory. Since the controller is $\overline{CAS}$ access (t_{CAC}) limited, the t_{CAC} spec of the 2118 and the 2164A must be compared for the read cycle. t_{CAC} on the 2164A-15 is 85 ns, 5 ns greater than the 2118. This means that valid data will arrive at the 8086 processor 5 ns later, for the worse case, using the 64K device. The read cycle timing analysis shows this is still well within the 570 ns requirement of the 8086. During the write cycle, two parameters were of concern in the 5 MHz system:

t_{DS} (data set-up before $\overline{CAS}$)

t_{CWL} (leading edge of write to trailing edge of $\overline{CAS}$)

Since the t_{DS} spec is the same for both devices (0 ns), the original timing analysis for this parameter is still valid and the 2164A fits. The t_{CWL} spec for the 2164A-15 is 40 ns. This is 60 ns less than the 2118-15, so that substituting the 2164A actually relieves a tight timing spot in this design. The additional delay added to control line paths due to larger input capacitances of the 2164A is accounted for in the 8203 specification (the 8203 is specified to directly drive four rows of 2118's, only two rows of 2164A's for this reason). After adding decoupling to meet the 2164A-15 requirements, the 2164A memory system is up and running, doubling memory size and reducing device count by one-half.

4.4 System Reliability

The majority of microcomputer systems are designed into applications where system failure ranges from irritating (such as a vending machine failure) to a financial loss (such as a double debit from an electronic teller machine). While these are not life threatening failures, reliability is important enough to be designed into the system.

A memory system is one of the system components for which reliability is important. Also it is one of the few system elements which can be easily altered to enhance its reliability. The inclusion of some additional hardware allows the CPU to keep check on the integrity of the data in memory. Figure 23 represents a five TTL chip solution that, when added to the 5 MHz design example, allows error detection in the memory.

Because the 16-bit 8086 has the ability to do selective high or low byte writes in addition to full word operations, parity needs to be generated and checked on the byte level. This requires two extra memory devices per row to store the parity bits of the high and low bytes.

Parity is generated by exclusive ORing all the data bits in each byte (accomplished by the 74S280) which results in a parity bit. This parity bit is the encoding bit of each byte. Because there are eight data bits, the parity bit C is: $C = b_1 \oplus b_2 \ldots . b_7 \oplus b_8$ where b = value in the bit positions.

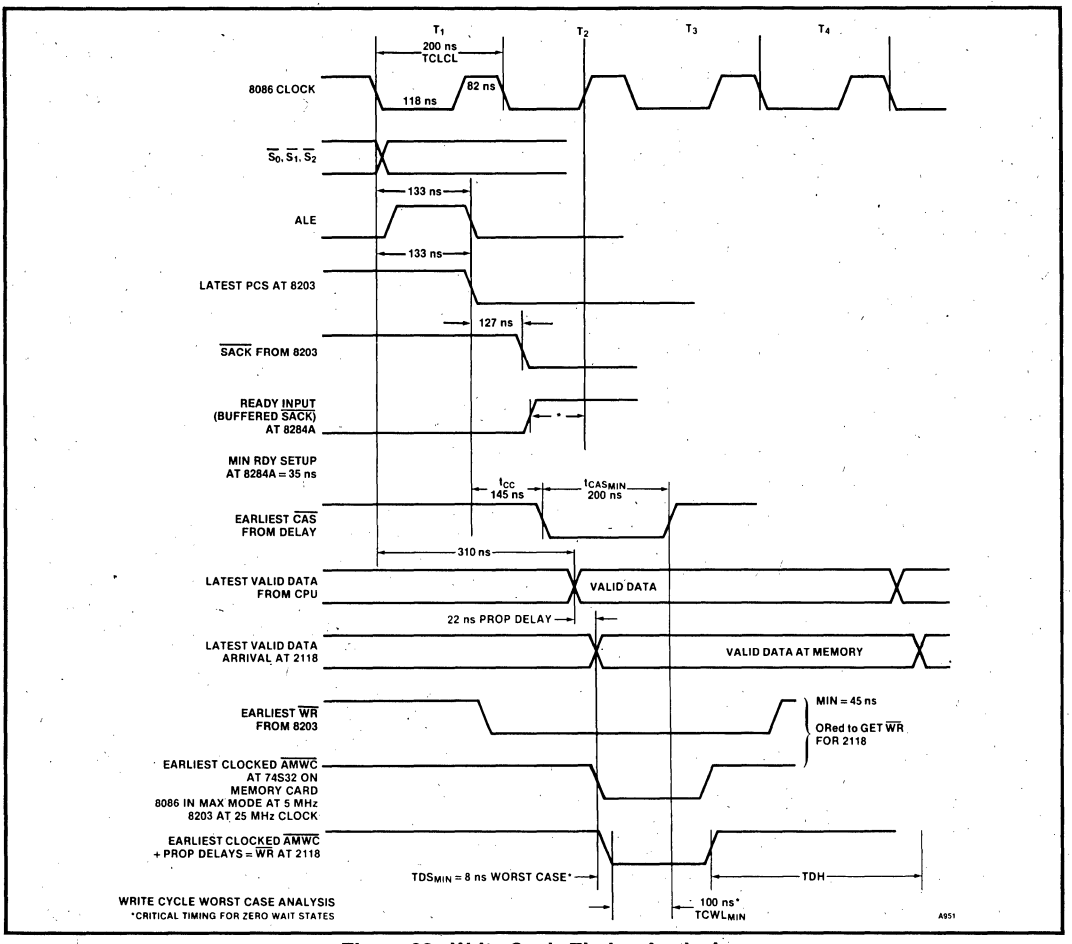


Figure 22. Write Cycle Timing Analysis

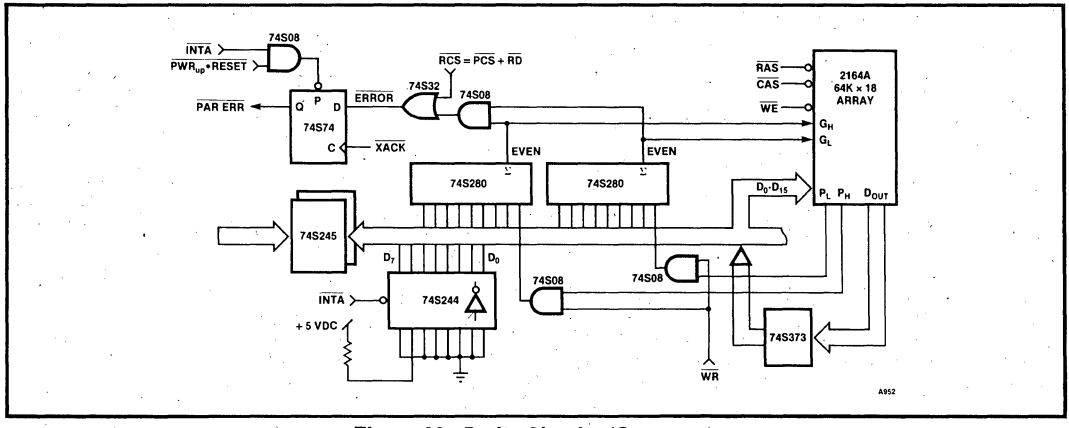


Figure 23. Parity Checker/Generator

The parity bit combines with the bits from the original data byte to form the encoded half-word (9-bit byte). Encoded words always have either "odd" parity, which is an odd number of 1s (an odd weight) or "even" parity which is an even number of 1s (an even weight). Odd and even parity are never intermixed, so that the encoded words have either odd or even parity — never both.

When the encoded word is fetched, the parity bits are removed from the word and saved. Two new parity bits are generated from each byte. Comparing these new parity bits with the stored parity bit determines if a single bit error has occurred in either byte.

Consider the two bit data word whose value is "01". Exclusive-NORing the two data bits generates a parity bit which causes the encoded word to have odd parity:

$C = \overline{0 \oplus 1}$

$C = 0$

The encoded word becomes:

Data	Generated Parity Bit
01	0

Assume that an error occurs and the value of the word becomes "110." Stripping off the parity bit and generating a new parity bit:

transmitted parity = 0

transmitted word = 11

New parity of transmitted word = $\overline{1 \oplus 1}$ = 1; generated parity ≠ transmitted parity.

Note that the error could have occurred in the parity bit and the final result would have been the same. An error in the encoding bit as well as in the data bits can be detected.

Although parity detects the error, no correction is possible. This is because each valid word can generate the same error state. Illustration of this is shown in Table 10.

Table 10. Possible Errors

Possible Correct Word with Parity	Single Bit Error
0 0 1	0 1 1
1 1 1	0 1 1
0 1 0	0 1 1

Each of the errors is identical to the others and reconstruction of the original word is impossible.

Parity fails to detect an even number of errors occurring in the word. If a double bit error occurs, no error is detected because two bits have changed state, causing the weight of the word to remain the same.

Using the encoded word "010" one possible double bit error (DBE) is:

1 1 1
└— Parity

Checking parity:

$C = \overline{1 \oplus 1} = 1$

The transmitted parity and the regenerated parity agree. Therefore the technique of parity can detect only an odd number of errors.

In the circuit of Figure 23, parity is generated and checked in the same devices — the 74S280 pair. Should a parity error occur in either the high or low byte (or both) the error flip-flop is set, causing an interrupt to the 8086 to occur. When the 8086 responds with INTA (interrupt acknowledge) the flip-flop is reset. INTA also enables the 74S244 which gates the interrupt number onto the data bus. The interrupt request signal to the CPU indicates a memory error has occurred. The nature of the interrupt procedure is heavily dependent on the user application, but typically ranges from retry or recovery routines to simply turning on the parity error light and proceeding.

One other software consideration for this circuit is the requirement to initialize all the memory to a known state. This initialization is needed to properly encode all the memory to even parity. This is typically done upon power-up by writing zeros into all memory locations prior to program storage.

In summary, single bit parity will detect the majority of errors, but cannot be used to correct errors. Using parity introduces a measure of confidence in the system. Should a single bit error occur, it will be detected.

For a detailed treatment of error detection and also techniques for error correcting, refer to Intel application notes AP-46, "Error Detecting and Correcting Codes Part #1," and Application Note AP-73, "ECC #2 Memory System Reliability with Error Correction."

4.5 Alternatives to 8203 Refresh Control Designs

There are essentially four choices available when selecting a technique for refresh control circuitry. These are:

Separate controller

CPU Hardware Control

CPU Software Control

Circuitry Internal to the RAM

Figure 24 is an implementation of a separate controller design. This is a typical non-LSI version that requires 11 TTL packages, an 8282A octal latch, a 3242 address multiplexer/refresh counter, two bidirectional bus drivers, an 8212 octal latch and two active delay lines.

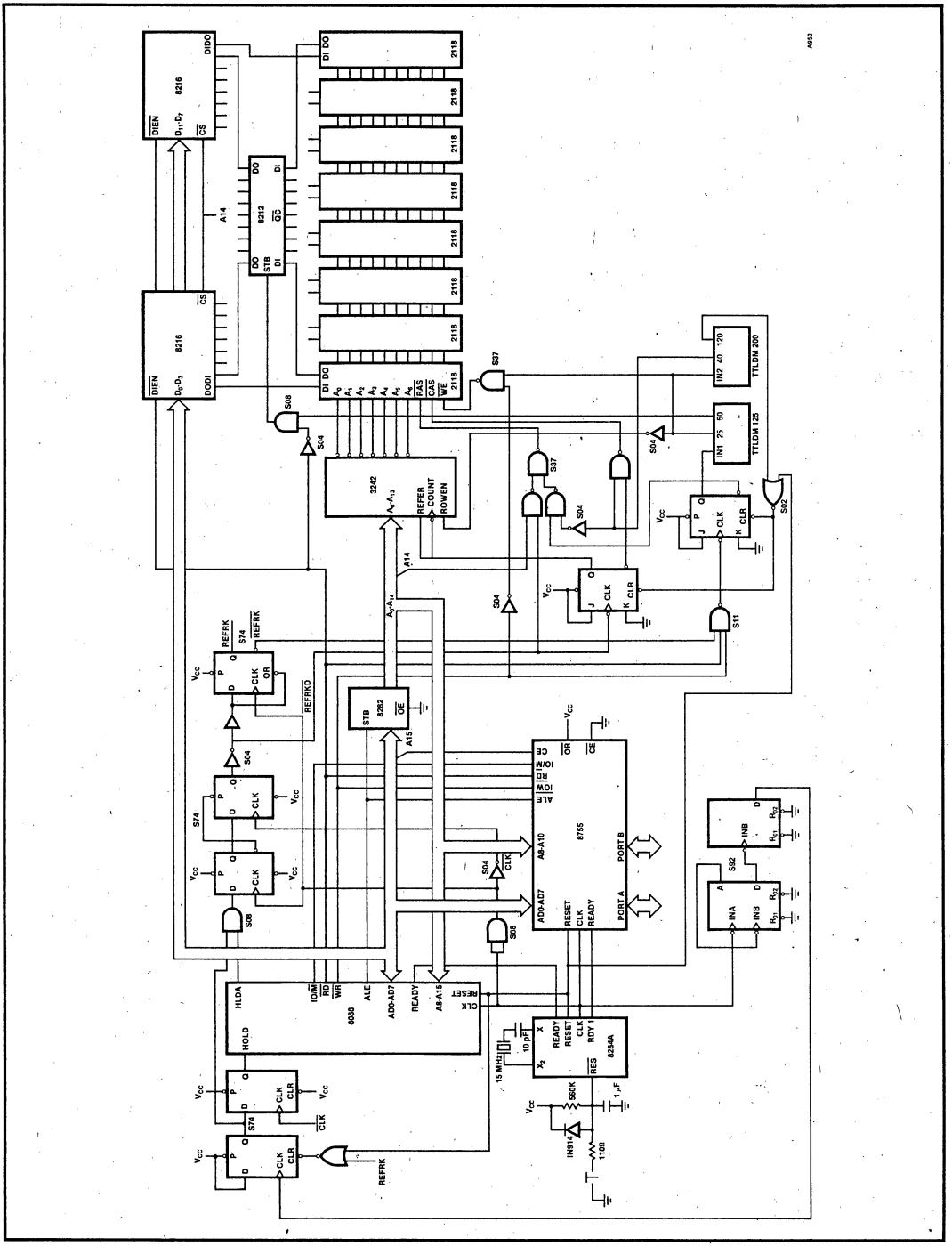


Figure 24. Discrete DRAM Controller

Nothing is gained by using discrete packages where a LSI device can be designed in. The plethora of TTL does require a larger engineering effort exemplified by the circuit complexity and timing analysis for this circuit (Figure 25). In terms of performance, the extra engineering effort can be fruitless — the CPU in this example is forced into the HOLD condition every time a refresh cycle occurs, even if the memory is not being accessed. This waiting period lasts 1.23 microseconds for every refresh cycle performed. Contrast this with the 8203 circuit which runs without WAIT states (unless a refresh cycle is in progress when the CPU requests a memory access, in which case one WAIT state is inserted). The advantages of using the 8203 should be obvious by now.

Additional hardware closely coupled to the CPU timing refresh for the microprocessor operation is one alternative to 8203 design. Some implementations include the extra hardware within the microprocessor; rendering a low cost, simple design. Wide restrictions govern the usage of such a system however, precluding this type of design in many applications.

To cite a few disadvantages:

- CPU must run continuously — no single step, HOLD, or extended WAIT states
- Multiprocessor operation is difficult
- CPU must always participate in memory operations

CPU software control of refresh is another alternative. This approach increases software development and maintenance costs and may not be offset by the very low or no hardware overhead for refresh. One method requires real-time analysis of all modules and possible directions of the program, with branch-to-refresh instructions included in all paths so that a refresh procedure is executed at least every 2 ms. An option on this technique requires a single interrupt time, which, when it times out, interrupts the CPU, causing it to revert to the burst refresh software routine.

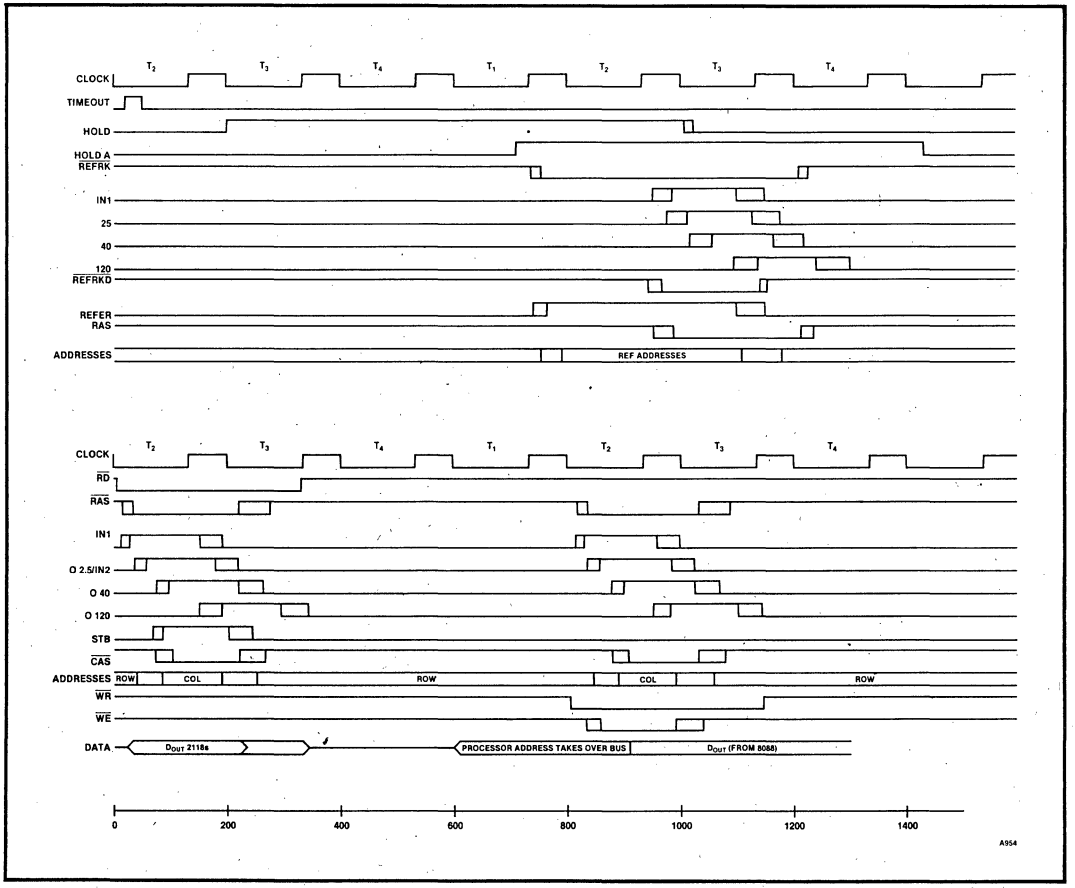


Figure 25. Timing Analysis Discrete Controller

Figure 26 shows an ASM-86 implementation of a burst refresh procedure. Accomplishing refresh in software is simple: save all registers used, perform a read at each of the 128 row addresses, then restore all registers and return.

The pure software approach makes it very difficult to make program changes and is limited to special applications. Also, since the refresh cycles are actually read cycles, the memory consumes more power for refresh than in standard refresh cycles. Both software refresh methods require that the CPU is always running, and hence shares many of the disadvantages of a CPU hardware refresh design.

One approach to memory system refresh control is to forge the entire system in silicon, incorporating the dynamic RAM array and all of the refresh control circuitry into one device. This, however, represents a departure from classical, dynamic RAM system design methodologies and as such, are outside the scope of this application note.

```
;*************************************************************
;
;            BURST REFRESH ROUTINE IN ASM86
;                  VERSION 1.0
;            MC APPLICATIONS LAB JAN 82
;
;*************************************************************

CSEG SEGMENT
ASSUME CS:CSEG,DS:CSEG

;************** BURST REFRESH INTERRUPT ROUTINE ***************

; This procedure does software refresh from an interrupt by
; performing dummy reads on the first 128 device (row)
; addresses.

; HARDWARE ASSUMPTIONS: RAS is common throughout the array
; with CAS decoded for a row select. An external timer
; generates the refresh interrupt every 2 milliseconds.

;*************************************************************

BURST:                          ;SAVE REGISTER CONTENTS
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    SI
        MOV     BX,BASEADRS     ;PLACE SEG PNTR OF TARGET BOARD ROW IN BX
BURS1:  MOV     DS,BX           ;INIT DATA SEG TO START OF A BOARD ROW
        MOV     CX,REFCOUNT     ;SET LOOP COUNTER TO NUMBER OF DEVICE ROWS
        MOV     SI,ADRCOUNT     ;INIT MEM INDEX PNTR
REF:    MOV     AX,DS:[SI]      ;READ 16 BIT WORD (DUMMY READ IS A REFRESH)
        DEC     SI
        DEC     SI              ;DECREMENT REFRESH ADDRESS PNTR TO NEXT WORD
        LOOP    REF             ;LOOP ONCE FOR EACH DEVICE ROW

; 128 ROWS HAVE BEEN READ, (REFRESHED) SO EXIT

EXIT:   POP     SI              ;RESTORE REGISTERS
        POP     CX
        POP     BX
        POP     AX
        IRET                    ;RETURN FROM INTERRUPT

BASEADRS EQU 0000   ;SET TO SEGMENT ADDRESS OF MEMORY
REFCOUNT EQU 128    ;SET TO NUMBER OF DEVICE ROWS (128 FOR 2118)
ADRCOUNT EQU 256    ;SET TO TWICE NUMBER OF DEVICE ROWS

CSEG ENDS

END
```

Figure 26. PLM-86/ASM-86 Burst Refresh, Sheet 1 of 2

```
REFRSH:
 DO;
  BURSTREF: PROCEDURE;     /* PROCEDURE PROVIDES A BURST REFRESH BY READING
                              ALL 128 DEVICE ROWS ON ALL BOARD LEVEL ROWS*/


  INCADR: PROCEDURE(PTR) POINTER;  /* INCREMENTS REFRESH ADDRESS POINTER */

   DECLARE  PTR POINTER,
            ADR BASED PTR (2) WORD;

   ADR(1)=ADR(1)+2;                /*INC WORD ADDRESS*/
   RETURN PTR;
  END INCADR;


  INCBD: PROCEDURE (PTR)POINTER;  /* INCREMENTS BOARD LEVEL ADDRESS POINTER */

   DECLARE PTR POINTER,
         ADDR BASED PTR (2) WORD;

   ADDR(0)=ADDR(0)+03FFFH;
   IF ADDR(1)=0 THEN ADDR(0)=ADDR(0)+1;
   RETURN PTR;
  END INCBD;


  DECLARE (BDROW$PTR,REF$PTR,START$PTR,LAST$PTR )POINTER;
  DECLARE (REF BASED REF$PTR,RDDATA ) WORD;
  DECLARE (DEVROWS) BYTE;


      /* READ 128 ADDRESSES ON ALL BOARD ROWS */

  DO;
   START$PTR=20000H;
   LAST$PTR=3FFF0H;
   BDROW$PTR=START$PTR;
   REF$PTR=START$PTR;
    DO WHILE BDROW$PTR<=LAST$PTR;
     DEVROWS=128;
      DO WHILE DEVROWS>=0;
       RDDATA=REF;
       REF$PTR=INCADR(REF$PTR);
       DEVROWS=DEVROWS-1;
      END;
     BDROW$PTR =INCBD(BDROW$PTR);
     REF$PTR=BDROW$PTR;
    END;
  END;

 END BURSTREF;

          /* MAIN */

 DO;
  CALL BURSTREF;
 END;

END REFRSH;
```

Figure 26. PLM-86/ASM-86 Burst Refresh, Sheet 2 of 2

One last technique for refresh control exists that doesn't fit into any of the above catagories and is worth bringing to light. Its use is heavily application dependent, hence has the most severe limitations, but if it meets the design requirements, its the most cost effective of all. The memory must be configured so that all row addresses will be strobed within 2 ms. Figure 27 is a block diagram of an application where this is possible since successive memory access addresses are predictable and defined. The circuit depicts a simpified graphics terminal display memory interface. Assuming a requirement of a 512×512 display resolution, the memory array is arranged as two rows of eight 2118 devices. During each read cycle, one byte is loaded from the memory into the shift register and is serially clocked out as video. A single $\overline{RAS}$ is common to the array and $\overline{CAS}$ is decoded to each row. This configuration simultaneously refreshes one row while reading data from the other row. A disadvantage of this arrangement is additional power supply and decoupling requirements, since one row is always making a transition to active current (ΔI_A) while the other draws refresh cycle current (ΔI_R). Refer to Section 6.3.4 on decoupling for calculations. The following is determined:

Pixel Clock (Hz) = (N + R) * L * F = 21.450 MHz

where N = Number of displayed dots per line = 512

L = Number of horizontal lines per frame = 532 (512 visible lines + 20 line times allowed for vertical retrace)

F = Frame rate of 60 Hz

R = Number of pixel clock times allowed for horizontal retrace time = 160 (Usually empirically determined. This number establishes the width of the margins on the left and right sides of the CRT display.)

Memory Cycle Rate = Byte read rate of the memory = 2.68 MHz

$$M_{cyc}\ (Hz) = \frac{21.450\ MHz}{8}$$

$$\frac{\text{pixel rate}}{\text{pixels/byte}} = 2.68\ MHz$$

$$T_{cyc} = \frac{1}{2.68\ MHz} = 373\ ns/cycle$$

The 2118-15 meets this T_{cyc} cycle time requirement.

Since the memory array is sequentially addressed, the memory is automatically refreshed every 128 consecutive cycles.

Checking refresh timings: 128 cycles × 373 ns/cycle = 47.74 microseconds between total refresh for each device, easily within the 2 ms specification.

The worst case refresh occurs during vertical retrace time when:

retrace time = 31.3 microseconds/line × 20 lines = 627 microseconds

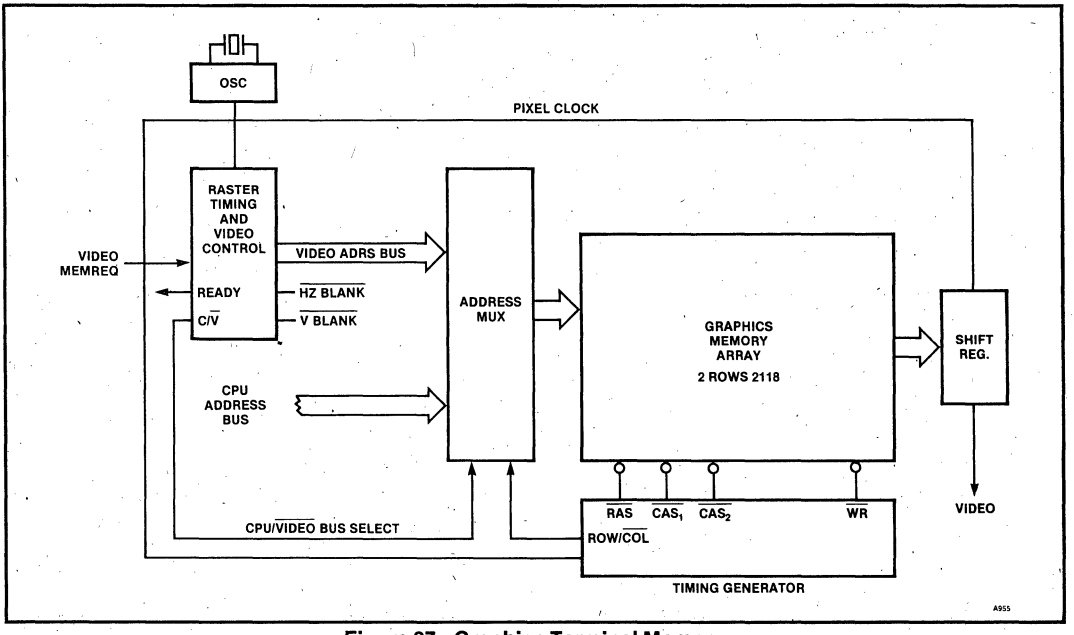


Figure 27. Graphics Terminal Memory

worst case refresh rate = 627 microseconds + 47.7 microseconds = 674.7 microseconds, still well within the 2 ms specification.

Writing is performed during horizontal or vertical retrace. More efficient designs would interleave memory, eliminating the processor being in WAIT mode until the memory is open. Here, and in some other limited applications, refresh can occur automatically by design, and with no software or hardware overhead.

5 10 MHz NO-WAIT STATE SYSTEM

For fast high performance microprocessors such as the 10 MHz 8086, an LSI controller for dynamic RAM interfacing is unacceptable, due to the requirement for WAIT states and resultant impact on performance. Until faster LSI controllers appear, discrete controller designs are required. In the example that follows, high performance design techniques are coupled with Intel high performance RAMs to yield a 10 MHz no-WAIT state 8086/2164A system.

The key requirements are:

ALE to data in:	219 ns
READY response:	89 ns
2164A t_{RAC}	150 ns

The solution and implementation that follows, configures the 8086-1 in max-mode, incorporates a synchronous arbiter while providing a quasi-synchronous refresh (refresh that is synchronous to the system clock, but not to the microprocessor).

5.1 System Refresh

Rather than being constrained to the design configurations of purely synchronous or asynchronous refresh arbitration, a quasi-synchronous scheme was chosen — taking advantage of the benefits of both, and avoiding some of the drawbacks of implementing either one exclusively. Synchronizing the refresh arbitration to the system clock ensures that its operations are inherently and closely coupled to CPU operation and allowing critical timing edges to always be predicted through worst case analysis. However, unlike totally synchronous systems, if the CPU in this example were to enter a HOLD, HALT, or otherwise stopped state, refresh cycles would continue to keep valid data in the memory, independent of the CPU operation. Also, synchronization of refresh requests to the system clock make the task of the arbiter very easy. Memory cycle requests and refresh cycle requests never occur at the same time (Figures 28 and 29, timing analysis). As a result, there is no chance that a random cycle request can arrive in a narrow time window that would violate data setup and data hold time of a flip-flop arbiter. This is a major problem in purely asynchronous designs.

5.2 System Block Diagram

Figure 30 is a block diagram of the basic functions required for this system; refresh interval timer, refresh address counter, arbiter synchronization, address multiplexing and timing generation. Included also in the diagram are the memory and CPU status decoders, data latches and transceivers, bus control and clock generation.

The function of the refresh interval timer is to place requests for refresh cycles, distributed in approximately 15 microsecond intervals, so that each row of the memory devices receives a refresh within 2 milliseconds. This timer is comprised of two four-bit synchronous binary counters and two flip-flops. The timer circuits divide the 10 MHz system clock by 150, then latches the count carry bit to hold until recognized, through the arbiter, by the refresh latch.

The refresh address counter generates the refresh addresses that are submitted to the address multiplexer during a refresh cycle. The counter is incremented once at the end of each refresh cycle to update the refresh address. The outputs are wire-ORed to the microprocessor address bus and are active only during a refresh cycle, at which time the current count is presented to the address multiplexer as the refresh address.

Timing generation for the memory array produces the control signals for the address multiplexer and the gating signals that provide for the properly timed arrival to the memory of $\overline{RAS}$, $\overline{CAS}$, and addresses. In this design example, it is essentially a delay circuit with variable taps to permit fine tuning of the memory inputs so as to allow no-WAIT states by the microprocessor for a memory cycle. The strobe used to latch valid data from the memory is also provided by the timing generator.

The 2164A dynamic RAM requirement of multiplexed row and column addresses is met by the address multiplexer. Here, the proper selection and transmission of row/refresh or column addresses is accomplished by control of the select line timing generation circuit.

In this design (Figure 31), arbitration is easily performed, i.e., once a cycle type is latched into its respective flip-flop (refresh latch or memory access latch) its request is presented to the input of an AND gate that will allow the request to pass through if a request of the other type is not currently in execution. Once the request passes the AND gate, the hardware is committed to a cycle of the requesting type and blocks any subsequent request until the current cycle is complete.

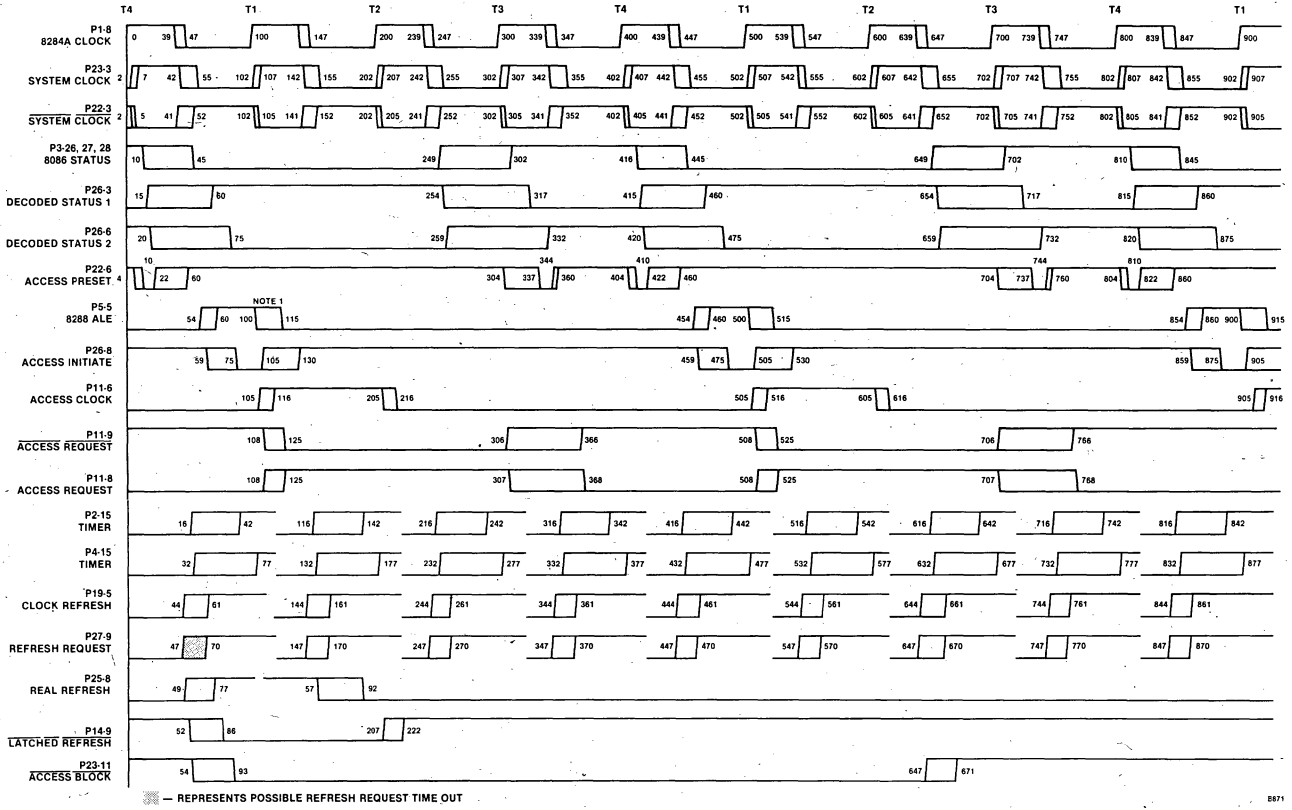


Figure 28. Refresh Cycle Followed by a Read/Write Cycle

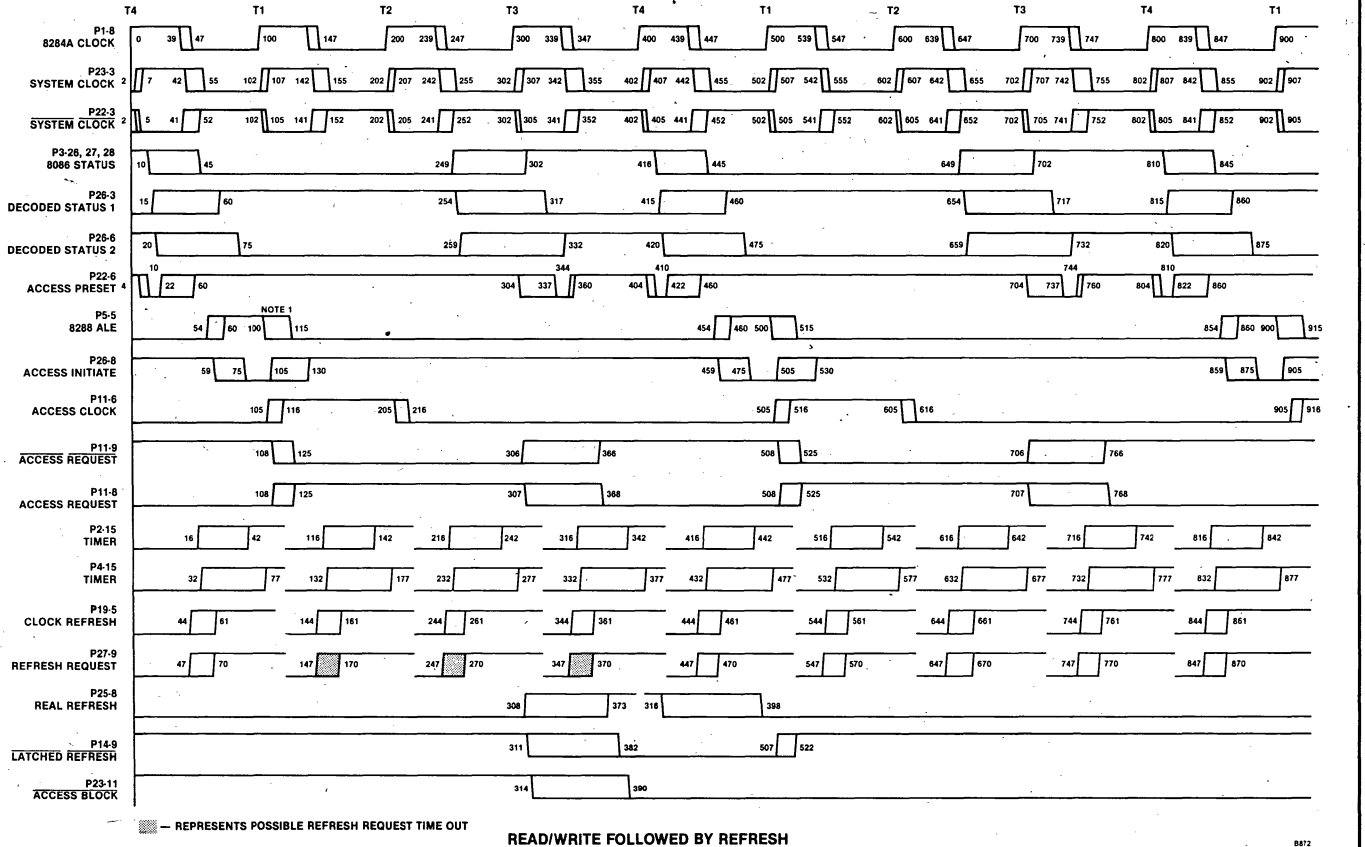


Figure 29. Read/Write Cycle Followed by a Refresh Cycle

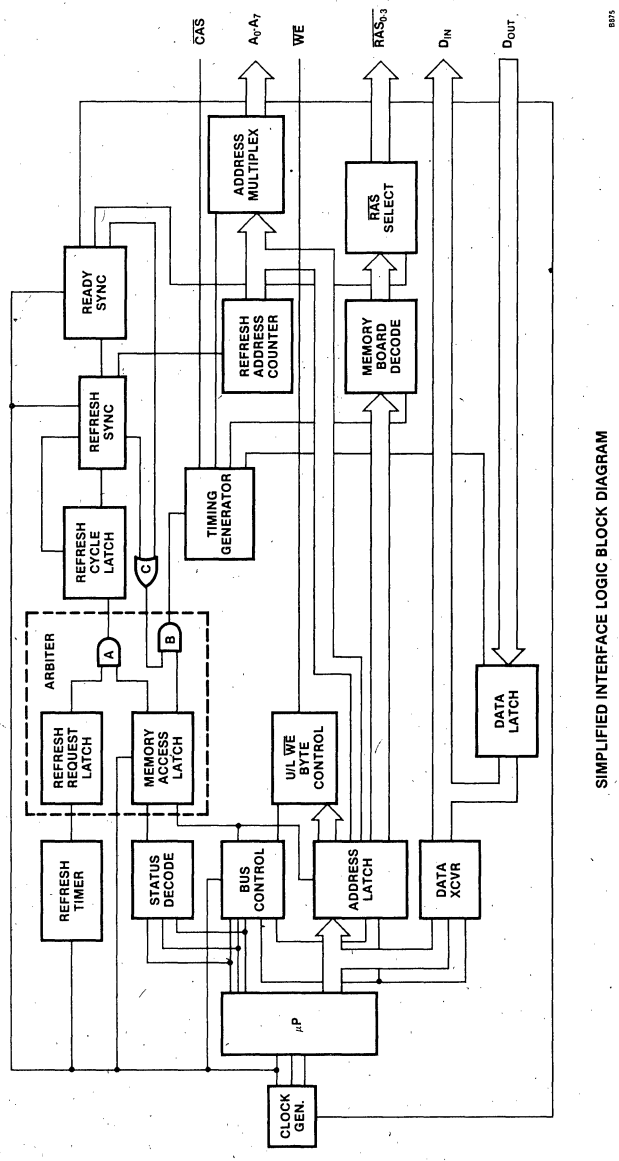


Figure 30. Simplified Interface Logic Block Diagram

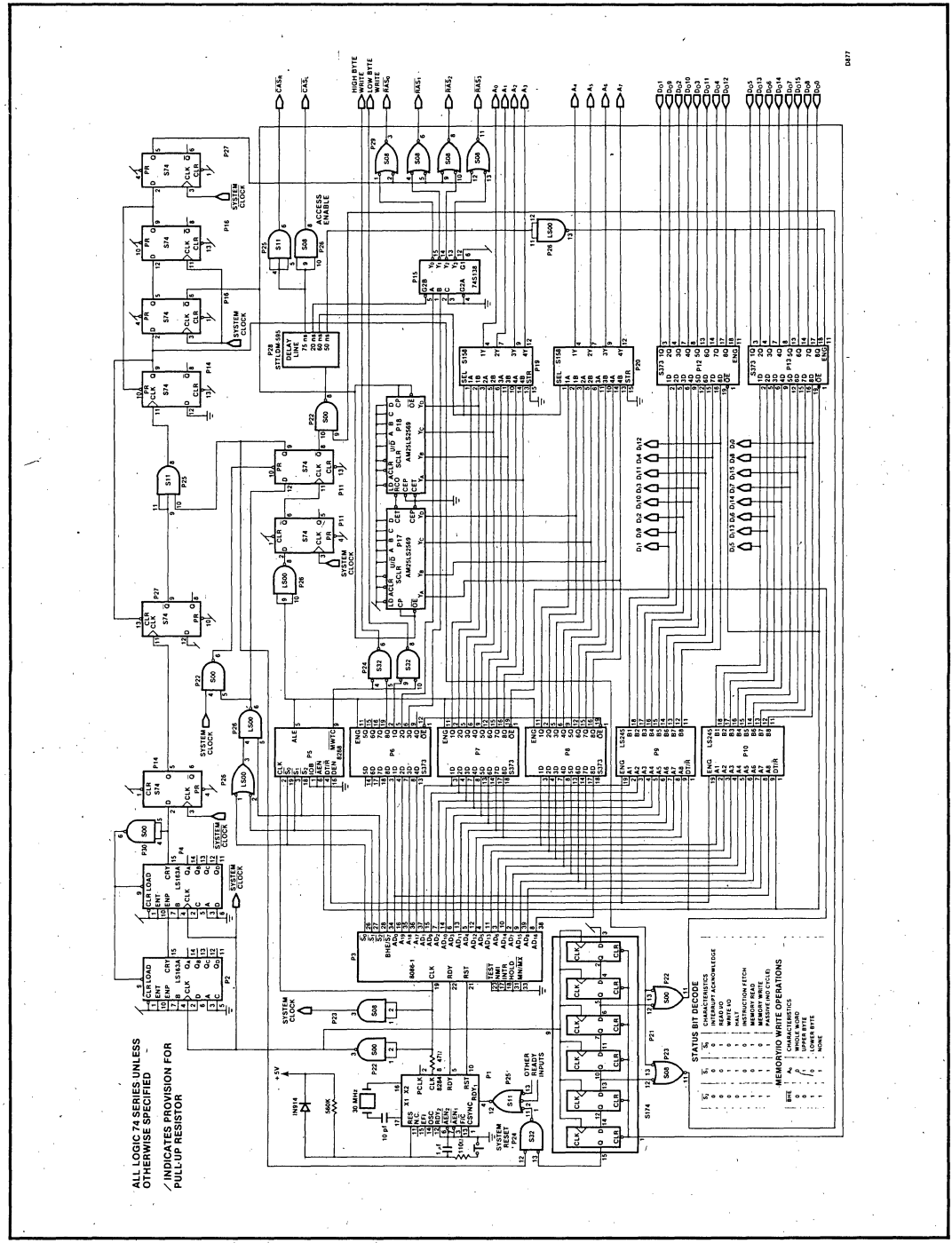


Figure 31. Logic Diagram — 10 MHz System

For example, suppose the CPU status decoder indicates a memory cycle is pending and there is no refresh cycle in progress. The status decoder outputs a bit indicating this condition to the memory access latch and is latched on the falling edge of ALE (address latch enable). After propagating through the latch, this latched memory access bit is presented to the input of AND gate B (where it will carry through the gate initiating a memory cycle, since there is no refresh cycle in progress) and its complement to AND gate A where it will block a refresh request from propagating through until the memory cycle is complete. As another example, assume that a refresh cycle is pending. The refresh timer times out, latches its output signal into the refresh request latch which subsequently presents this latched refresh request to the input of AND gate A. Here the signal is either held up or passed through depending upon the current CPU status. Assuming that there is no memory cycle in progress or that one has just ended, the AND gate passes the refresh request through to the refresh cycle latch, committing the hardware to initiate a refresh cycle and blocking any memory request that may occur until the end of the refresh cycle.

The sole purpose of the CPU status decode block is to inform the arbiter (as soon as possible) as to whether or not the pending machine cycle is going to be a memory cycle.

The bus controller provides the memory write command ($\overline{MWTC}$) and is steered to a high and/or low byte write by A0 and $\overline{BHE}$ in the byte control block. Address latch enable (ALE) used for latching valid addresses off the multiplexed bus, data enable ($\overline{DEN}$) used to enable the data transceivers, and data transmit/receive (DT/$\overline{R}$) used to control the direction of the data transceivers, are all provided by the bus control block.

The refresh sync and ready sync blocks generate several control signals for a number of functions that must execute to carry a refresh cycle to its natural end, all in synchronization with the system clock. The first signals generated are address disable — used to switch the CPU address latches into a high impedance state, and access block — used to block a memory cycle request at AND gate B. On the next rising clock edge a control signal is output that will switch the refresh address counters onto the address bus and enable a string of shift registers that comprise the ready sync to start shifting the READY bit through. Then, on the next rising edge of the clock, the refresh cycle latch is cleared, and finally on the falling edge of the clock the refresh signal is output from the ready sync block which is used by the $\overline{RAS}$ select block to enable all the $\overline{RAS}$ lines at once, simultaneously performing refresh on all four memory rows.

5.3 Schematic

Refer to the logic schematic (Figure 31) and to the block diagram in Figure 30, during the following discussion involving the conversion of logic blocks to TTL logic.

The refresh interval timer is comprised of devices P2 and P4, two 74LS163 four-bit synchronous binary counters, and one F/F from P14, a 74S74 flip-flop. The counters are cascaded and free-running, being incremented by the system clocks so as to output a refresh request pulse every 15 microseconds. This pulse is stored by P27 F/F, the refresh request latch, which is part of the arbiter.

The refresh counter is a pair of AM25LS2569 three-state binary up/down counters (located at P17 and P18) that sequence from 0 to 2^8-1 (255) and then roll over to start again. The MSB (most significant bit) of the counter is unused. With the devices' clock input tied to their $\overline{OE}$, the counters are automatically incremented at the end of a refresh cycle when the outputs are switched off the address bus by $\overline{OE}$ going high. This sets up the count to the next refresh address.

Memory address multiplexing is comprised of a pair of 74S158 quad 2:1 multiplexers (P19, P20). Inverted data output devices were selected because of their shorter propagation delay. The arrival of addresses to the memory is one of the tight timing constraints for zero WAIT states. The select line is controlled by the timing generator during a memory read or write cycle and is used to switch from row to column addresses at the appropriate time. During a refresh cycle, the select line does not change; thus, only the refresh addresses, which are wire-ORed to the row addresses are presented to the memory array.

The arbiter in this system is designed with two 74S74 F/Fs, one from P11 and the other from P27, and two gates: a 74S11 AND gate at P25 and a 74S00 NAND gate at P22. As previously discussed, the arbiter makes the decision of whether to run a memory R/W cycle or a refresh cycle, then commits the hardware to initiate the cycle decided upon. Classically a difficult choice, the task is greatly simplified by the quasi-synchronous nature of this design. Memory and refresh cycle requests never occur at or near the same time and the worst case data setup and hold times at each F/F are easily predicttable and are designed to avoid violations of these specifications. The relatively simple nature of this arbitration circuit is demonstrated by the small device count and simplicity of the method involved.

The status decode block is implemented with two NAND gates from 74S00 at P26 and one NAND gate from P22. Low power Schottky devices were required because of the limited (2 mA) drive capability of the 8086 status

lines. Through observation of the truth table for the status bits S0-S2 on the schematic and the following logic, it is apparent that NAND gate P26, pin 6 goes low during memory read, memory write, or instruction fetch cycles. This active low memory cycle status bit is latched into the access latch on the trailing edge of the clocked ALE (from S74 F/F at P11) and informs the arbiter that this memory cycle is in progress. For any other type of CPU cycle, device P26, pin 6 is high, which enables NAND gate P22, pin 5 to allow the next rising edge of the clock to preset the memory access latch, indicating to the arbiter that this is not a memory cycle.

The bus control block functions are executed with an Intel 8288 bus controller. In this circuit, ALE, DT/$\overline{R}$, $\overline{DEN}$ and $\overline{MWTC}$ are all generated at P5 from system clock and CPU status bits inputs. The $\overline{MWTC}$ is used for the write pulse to the memory array, being directed to the higher or low byte by the pair of 74S32 gates at P24 which comprised the U/$\overline{L}$ $\overline{WE}$ byte control block. ALE is transmitted to P11 latch control (ENG) input of the 74S373 three-state address latches P6-P8, thus latching valid addresses from the multiplexed CPU bus. DT/$\overline{R}$ and $\overline{DEN}$ are wired to pins 1 and 19 respectively of the pair of 8-bit 74LS245 data transceivers at P9 and P10, with DT/$\overline{R}$ controlling the direction of data flow through the devices and $\overline{DEN}$ used to enable the device output drivers in the direction selected by DT/$\overline{R}$.

Timing generation for memory array related signals are all derived from a STTLDM-595* active delay line at P28. Activated only during a memory cycle via a single input from the arbiter, this one pulse is delayed 25 ns to become the ACCESS ENABLE signal (the source of $\overline{RAS}$), 50 ns to enable the flow through memory data latches, 60 ns before switching the address multiplexer and finally delayed 75 ns before becoming the source of $\overline{CAS}$.

The ACCESS ENABLE line is connected to P5 of the 74S138 three-to-eight decoder located at P15. Configured as a two-to-four decoder by grounding the C-input and placing high order addresses A17 and A18 on the A and B inputs, P15 selects which of the four memory rows will receive a $\overline{RAS}$ signal. Once a proper output is selected, the ACCESS ENABLE signal is directed through the 74S138 to the correct row after being buffered through a 74S08 at P29. Note that one input of all the gates at P29 $\overline{RAS}$ buffers are connected together to the refresh signal. This allows simultaneous strobing of all memory $\overline{RAS}$ during a refresh cycle.

It is evident from the examples presented that the Intel 2118 and 2164A high performance DRAMs match any speed microprocessor memory requirement, fulfilling the needs at all performance levels. In particular, the 2164A DRAMs used in this 10 MHz design easily conform to the rigid requirements of this high performance system.

6 HIGH PERFORMANCE SYSTEM DESIGN CONSIDERATIONS

Designing a high performance, high speed memory system requires consideration of the following areas:

1. Skew
2. Propagation Delay
3. General Circuit Design Techniques
4. Worst-case timing analysis

6.1 Skew

Skew is the difference between maximum and minimun propagation delay through devices in a parallel path. For example, refer to Figure 32. Here signal A and signal B propagate through the same number and types of gates, each transversing a parallel path. For both signals the total mimimum delay is 6 ns and the total maximum delay is 16 ns. However, diagramming the worst case (Figure 33), the skew between these signals can be as much as 10 ns. This time (skew) adds directly to the system access/cycle time.

Capacitive loading of the STTL drivers will cause rise time degradation in the memory array, and will contribute to skew, caused by heavily loaded versus lightly loaded signals. Figure 34 displays the effects of capacitive loading of the Schottky TTL. Obviously skew needs to be minimized.

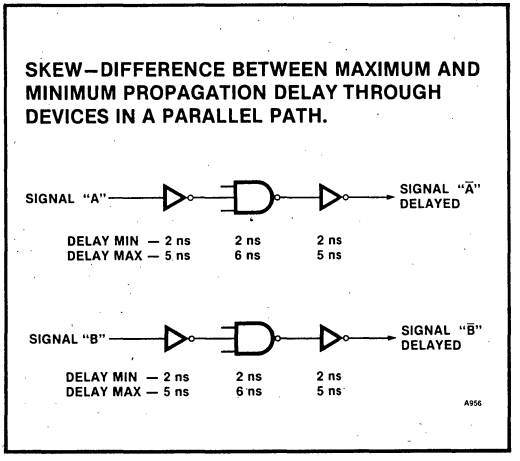


Figure 32. Skew — Variations Between Max/Min Propagation Delay

* Available from EC², San Luis Obispo, California

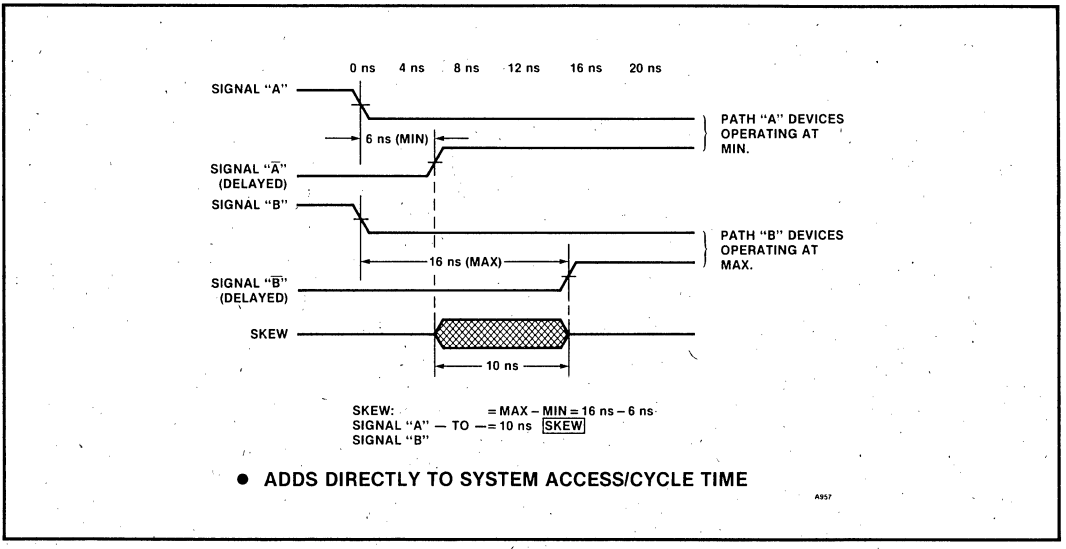


Figure 33. Skew — Adds Directly to System Access/Cycle Time

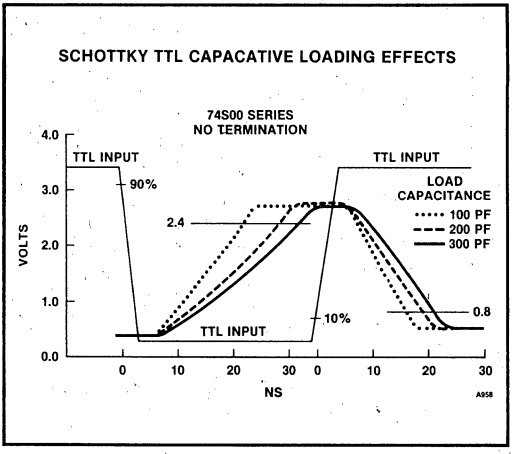


Figure 34. Schottky TTL Capacitive Loading Effects

The goal to minimize skew is achieved by observing the following guidelines:

- Select logic gates for minimum delay per function
- Place parallel paths in the same package (Device to device skew within the same package = .5 ns max for STTL, 2.0 ns max for high current drivers, i.e., 74S240.)
- Balance the output loading to equalize the capacitive delays
- Use delay lines with tight t_{prop} and t_{rise} tolerances (± 1 ns)
- Drive address and clocks from a common area on the P.C.B. to avoid circuit board trace skew due to unequal lengths of signal distribution (Figure 35).
- Localize the timing generation

6.2 Propagation Delay

Propagation delay must be determined in the critical paths to guarantee the design goals of circuit optimization and maximum performance. The following rules are generally used to determine propagation delay through the TTL devices:

- t_{prop} MAX = Data Book maximum
- t_{prop} Typical = Data Book typical
- t_{prop} MIN = ½ Data Book typical

Capacitive loads add to the propagation delays specified in the data books. The additional delay can be calculated in the following manner:

- Additional Delay = $D_C \times (C_{load} - C_{spec})$, where
  C_{load} = sum of all input capacitance plus PCB traces (≈ 2 pF/in),
  C_{spec} = specified capacitance of the driver, and
  D_C = the derating factor for the driver logic family
  - — Schottky TTL = 0.5 ns/pF
  - — Low power Schottky TTL = .1 ns/pF
  - — High current Schottky TTL = .25 ns/pF
  - — TTL = .75 ns/pF

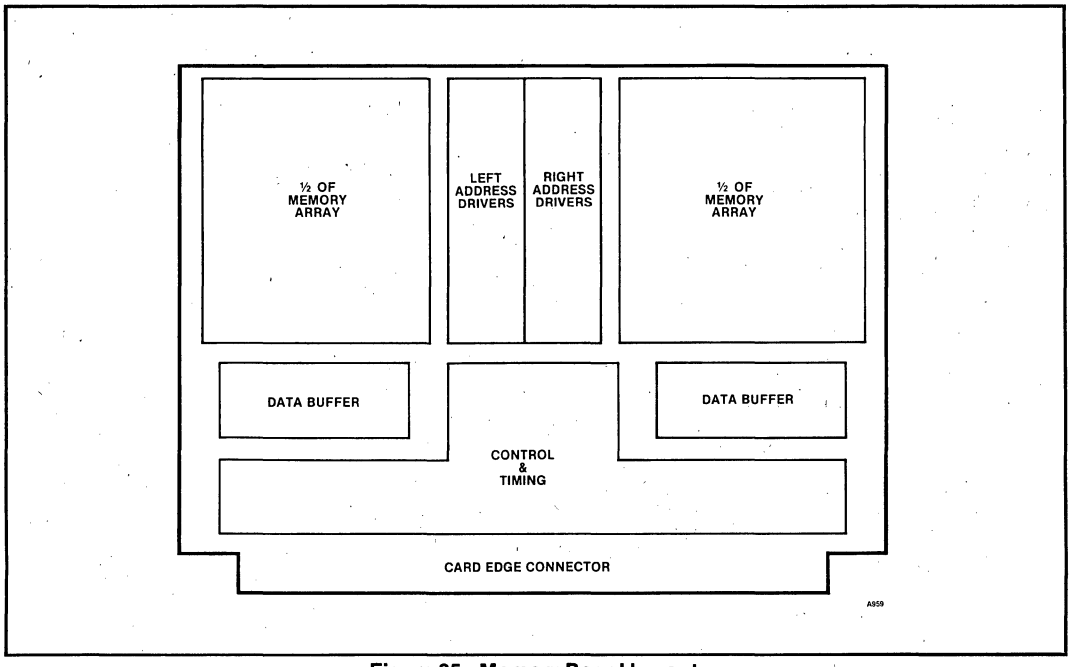


Figure 35. Memory Board Layout

6.3 Circuit Design Techniques

Optimum circuit design demands attention to the physical details of a 2164A memory system. A properly produced layout will minimize board area while yielding wider operating margins on timing and power supply requirements. The key areas of consideration are:

1. Ground and power gridding
2. Memory array/control line trace routing
3. Control logic centralization
4. Power supply decoupling

6.3.1 GROUND AND POWER GRIDDING

The power and ground network do not appear as a pure low resistance element, but rather as a transmission line because the current transients created by the RAMs are high frequency in nature. The RAMs are the lumped equivalent circuits of the power and ground transmission lines are shown in Figure 36.

The characteristic impedance of a transmission line is shown in Figure 37A. By connecting two transmission lines in parallel, the characteristic impedence is halved. The result is shown in Figure 37B.

Transient effects can be minimized by adding extra circuit board traces in parallel to reduce interconnection inductance.

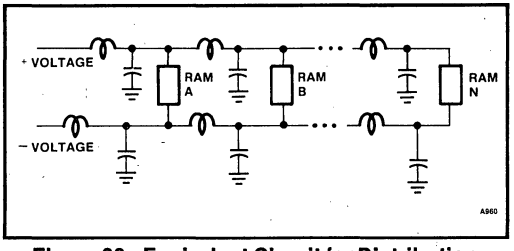


Figure 36. Equivalent Circuit for Distribution

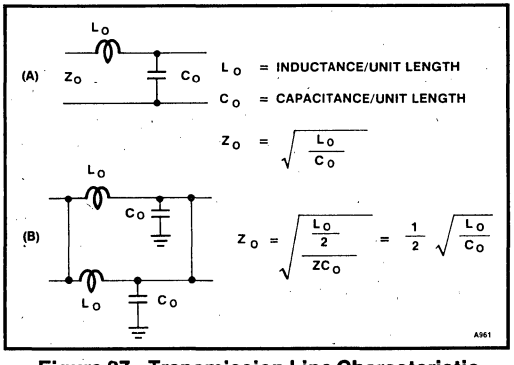


Figure 37. Transmission Line Characteristic Impedance

Extrapolation of this concept to its limit will result in an infinite number of parallel traces, or an extremely wide low impedance trace, called a plane. Distribution of power and ground voltages by plane provides the best distribution, however correct gridding can effectively approximate the benefits of planar distribution by surrounding each device with a ring of power and ground (Figure 38).

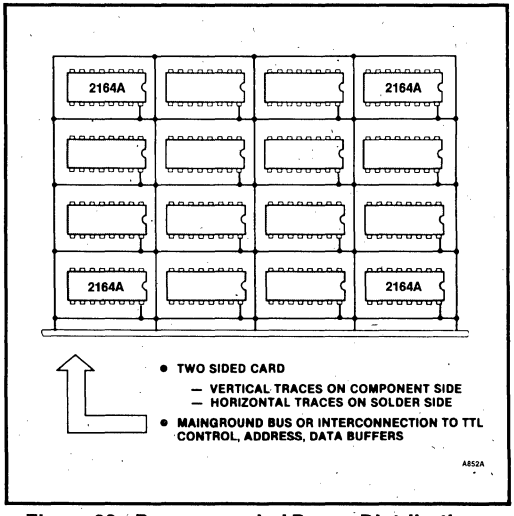


Figure 38. Recommended Power Distribution — Gridding

Improper ground and power gridding can contribute to excess noise and voltage drops if not properly structured. An example of an unacceptable method is presented in Figure 39. This type of layout promotes accumulated transient noise and voltage drops for the device located at the end of each trace (path).

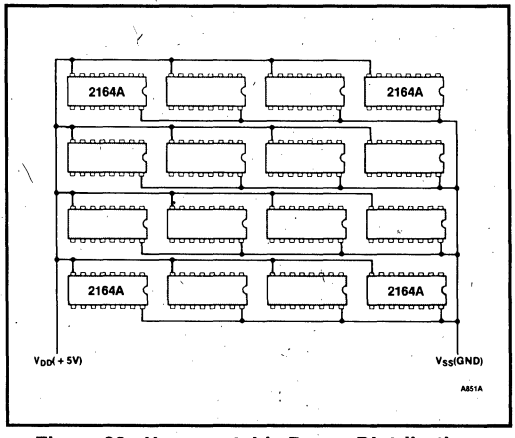


Figure 39. Unacceptable Power Distribution

6.3.2 MEMORY ARRAY/CONTROL LINE ROUTING

Address lines need to be kept as short and direct as possible. The lone serpentine line depicted in Figure 40 should be avoided, since the devices farthest away from the driver will receive a valid address at a later time than the closer ones. A better way to route address lines is in a comb like fashion from a central location as depicted in Figure 41. Routing control and address signals together from a centralized board area will also minimize skew.

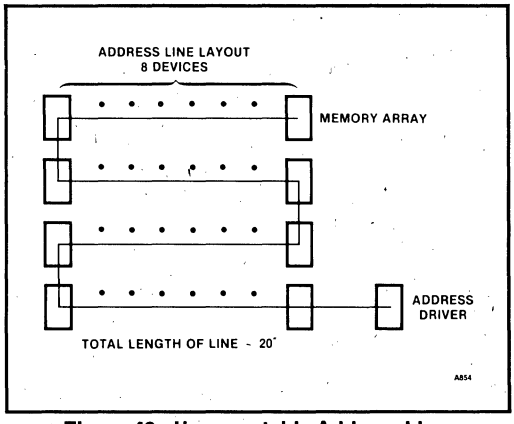


Figure 40. Unacceptable Address Line Routing (Serpentine)

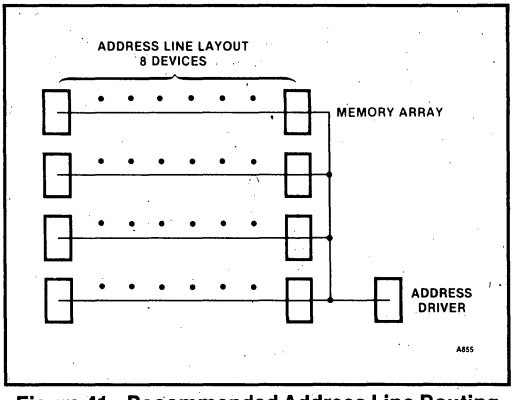


Figure 41. Recommended Address Line Routing

Allow for proper termination of all address and control lines, since a P.C.B. trace becomes a transmission line when:

$2t_{pd} \geq t_r$ or t_f

where: t_p = propagation delay down the line
t_r = rise time
t_f = fall time

The maximum unloaded line lengths not displaying transmission line characteristics are listed in Table 11. The values assume propagation delay of $\delta = 1.7$ ns/ft.

Table 11. Transmission Characteristics

Logic Family	Rise Time	Fall Time	Max. Length
54/74L	14 - 18 ns	4 - 6 ns	14.1 inches
54/74	6 - 9 ns	4 - 6 ns	14.1 inches
54H/74H	4 - 6 ns	2 - 3 ns	7.0 inches
54LS/74LS	4 - 6 ns	2 - 3 ns	7.0 inches
54S/74S	1.8 - 2.8 ns	1.6 - 2.6 ns	5.6 inches
10K ECL	1.5 - 2.2 ns	1.5 - 2.3 ns	5.3 inches
100K ECL	0.5 - 1.1 ns	0.5 - 1.1 ns	1.8 inches

The maximum length of a loaded transmission line is:

$$L_{max} = \sqrt{\left(\frac{C_D}{C_O}\right)^2 + \left(\frac{t_R \text{ or } t_F}{\delta}\right)^2} - \frac{C_D}{2C_O}$$

where C_D = Capacitive load/unit length
and C_O = Capacitance/unit length

6.3.3 CONTROL LOGIC CENTRALIZATION

Memory control logic should be strategically located in a centralized board position to reduce trace lengths to the memory array (Figure 35).

Long trace lines are prone to ringing and capacitive coupling, which can cause false triggering of timing circuits. Short lines minimize this condition and also result in less system skew.

A practical memory array layout is presented in Figure 42. Typically, this pattern and its "mirror image" are placed on each side of the memory control logic for a practical memory board design.

6.3.4 POWER SUPPLY DECOUPLING

For best results with the 2164A, decoupling capacitors are placed on the memory array board at every device location (Figure 42). High frequency 0.1 μF ceramic capacitors are the recommended type. In this arrangement each memory is effectively decoupled and the noise is minimized because of the low impedence across the

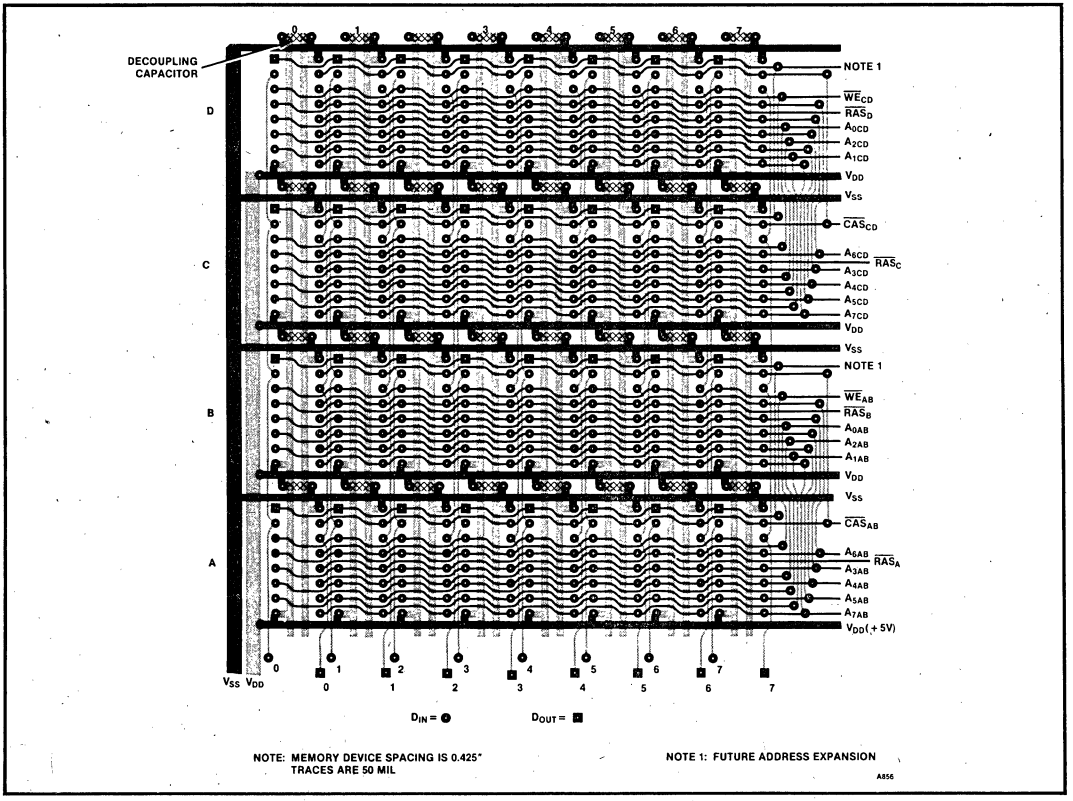


Figure 42. 2164A Memory Array P.C. Board Layout

circuit board traces. Typical V_{DD} noise levels for this array are less than 300 mV.

A large tantalum capacitor (typically one 100 μF per 32 devices) is required for the 2164A at the circuit board edge connector power input pins to recharge the 0.1 μF capacitors between memory cycles.

Decoupling is of considerable importance in circuit design in order to minimize transient effects on the power supply system. In order to determine the values for proper decoupling capacitors, the required amount of charge storage for a capacitor must first be determined in the following manner:

$Q = \Delta I \Delta T$

where: Q = charge in coulombs
ΔI = change in current is amperes
ΔT = change in time is seconds

and: $\Delta V = Q/C$

where: ΔV = voltage change in volts
and C = capacitance in farads

Assuming the following system parameters:

5 mA to 55 mA current switch for regular cycle
5 mA to 45 mA current switch for refresh cycle
1 microsecond bulk decoupling response time
260 ns cycle time
¼ of devices selected (one of four rows)

An example calculation proceeds as follows:

Q = (45-5 mA) (.3 μsec) + ¼ (55-5) mA (.7 μsec)
Q = 20.75 nanocoulombs

if V_{DD} is restricted to 100 mV (2%) then

$$C = \frac{20.8\text{ nC}}{100\text{ mV}} = .21\ \mu\text{F/Device}$$

if V_{DD} is allowed to 500 mV (10%) then

$$C = \frac{20.8\text{ nC}}{500\text{ mV}} = .042\ \mu\text{F/Device}$$

Bulk decoupling requirements are determined in a similar way:

Assuming the following:

50 μsec power supply response time
15.6 μsec refresh rate
Three refresh cycles/50 μsec period
I_{DD} standby = 5.77 mA

$$I_{DD}\ \text{STDBY} = \frac{45\text{ mA (.3 }\mu\text{sec)} + 5\text{ mA (15.3 }\mu\text{sec)}}{15.6\ \mu\text{sec}}$$

$$= 5.77\text{ mA}$$

An example calculation with ¼ devices active proceeds as shown:

$$Q = [50 - (3)(.3)]\ \mu\text{sec} \times 49.23\text{ mA } (¼) = 604\text{ nC}$$

$$\text{if V} = 100\text{ mV then C} = \frac{604\text{ nC}}{100\text{ mV}} = 6.0\ \mu\text{F device}$$

$$\text{if V} = 500\text{ mV then C} = \frac{604\text{ nC}}{500\text{ mV}} = 1.2\ \mu\text{F device}$$

The data shown in Table 12 defines the decoupling requirements of 2164A-15 and 2118-15 dynamic RAMs for a 300 ns cycle time over various device selections for a given percentage.

Cycle time has a downward scaling effect on the average operating current according to the following equation:

$$I_{DD_{AVE}} = \left[I_{DD2} \times \left(\frac{t_{RC}\text{ (spec)}}{t_{RC}\text{ (operating)}} \right) \right] + \left[I_{DD1} \times \left(1 - \frac{t_{RC}\text{ (spec)}}{t_{RC}\text{ (operating)}} \right) \right]$$

At minimum cycle time, $\frac{t_{RC}\text{ (spec)}}{t_{RC}\text{ (operating)}} = 1$,

so that worst case $I_{DD_{AVE}} = I_{DD2}$, but as the cycle time increases, $I_{DD_{AVE}}$ approaches the standby current,

Table 12. Decoupling Chart

	% Selected Devices	$\Delta V_{DD} = 2\%$		Cycle Time	$\Delta V_{DD} = 10\%$		
		C_D	C_B		C_D	C_B	
2164A-15	100	0.47	24.0	300 ns	0.11	4.8	0.1 μfd/device will work if ¼ devices are active at one time. + 100 μfd every 32 devices.
	50	0.29	12.0	300 ns	0.059	2.4	
	25	0.21	6.0	300 ns	0.042	1.2	
	12.5	0.16	3.0	300 ns	0.033	0.6	
2118-15	100	0.19	9.2	300 ns	0.038	1.84	0.1 μF/2 devices + 27 μF every 32 devices (assuming ¼ of devices active)
	50	0.10	4.6	300 ns	0.019	0.92	
	25	0.064	2.3	300 ns	0.013	0.46	
	12.5	0.048	1.15	300 ns	0.01	0.3	

becoming 6.3 mA @ 10,000 ns cycle time. Figure 5 in the 2164A data sheet depicts this scaling effect. Be sure to use the correct I_{DD} value based on specific worst case cycle time when computing specific decoupling requirements.

6.4 Timing Analysis — Determining the Worst Case

Once the control logic is designed, worst case system delays must be determined to guarantee proper circuit operation. There are two ways to perform these calculations:

1. A statistical worst case analysis (or the Monte Carlo method) which assumes that all devices probably won't be in their worst case condition at the same time.

   It is determined by the following formula:

   STATISTICAL WORST CASE

   $= \sqrt{\Sigma(A)^2 + (B)^2 + (C)^2}$ MAX STTL DELAYS + TYPICAL STTL DELAYS

   $+ \sqrt{\Sigma (A)^2 + (B)^2 + (C)^2}$ SKEW DELAYS

   + Σ DELAYS DUE TO CAPACITIVE LOADING

   + *MAXIMUM* DELAY ACCESSING MEMORY DEVICE

   WHERE (A), (B) OR (C) = MAX-TYP OR TYP-MIN

2. A true worst case analysis, using specified maximum and minimum delays for peripheral circuits plus all delays due to capacitive loading from device inputs and distributive capacitance in PC board etched con ductors. The following formula appears here:

   WORST CASE

   = Σ MAX STTL DELAYS + SKEW DELAYS (PERIPHERAL DEVICES)

   + Σ DELAYS DUE TO CAPACITIVE LOADING (INPUTS + P.C.B. TRACES)

   + *MAXIMUM* DELAY ACCESSING MEMORY DEVICE (T_{RAC} OR T_{CAC})

Since the statistical approach can be justified only in large systems with hundreds or thousands of components, the timing calculations used in all of the previous examples are based on a true worst case analysis. Capacitive delay is formulated from the equations in Section 6.1.2.

In summary, the following rules and guidelines apply to worst case analysis:

1. All propagation delays are from the industry TTL books.

   Max = Data book maximum

   Typ = Data book typical

   Min = ½ Data Book Typical

2. Skew device to device in same package = 0.5 ns Max for Schottky TTL and 2 ns for 74S240.
3. STTLDM-595 is a special delay line with active outputs. Propagation delay = ± 1 ns per tap (i.e., 75 ± 1 ns). (10 MHz system.)
4. Capacitive loads add 0.5 ns/pF to propagation delays specified in device spec (i.e., 74S04 is specified at 5.0 ns Max @ 15 pF. At 25 Pf propagation delay is 5.5 ns) Schottky TTL input capacitance is 3 pF. PCB traces are 2 pF/inch.
5. PCB etch delay adds little or no skew to array address/control timing signals. It adds 4 ns, however, in the overall access time data path.
6. Timing components are immediately adjacent to each other, making PCB etch delays in delay timing chain negligible (exception is timing tap used to terminate delay line latch).

7. SUMMARY

The Intel 2164A and 2118 DRAMs meet all microprocessor system requirements, offering high density, speed, low power and ease of use. Follow the system design guidelines presented to create a harmonious microprocessor memory design.

8. REFERENCES

AP-75	Application of the Intel 2118 16K Dynamic RAM
AP-131	2164A 64K Dynamic RAM Device Description
AP-92A	Interfacing Dynamic RAMs to iAPX 86/88 Systems Using the Intel 8202A and 8203
AP-46	Error Detecting and Correcting Codes Part #1
AP-73	ECC #2 Memory System Reliability with Error Correction